SP-1764

Designing an Artificial Student

Aiko Hormann

7 October 1964

SP-1764

SP *a professional paper*

Designing an Artificial Student[*]

by

Aiko Hormann

7 October 1964

SYSTEM

DEVELOPMENT

CORPORATION

2500 COLORADO AVE.

SANTA MONICA

CALIFORNIA

[*]Presented at the International Meeting of the Operations Research Society of America (Western Section), Honolulu, Hawaii, September 15, 1964.

SDC

ABSTRACT

This paper describes the design philosophy and conceptual scheme for Gaku--a system of computer programs that is capable of exhibiting learning and problem-solving behavior. The concrete manifestation of this system consist of four mechanisms--a programming mechanism, a problem-oriented mechanism, a planning mechanism, and an induction mechanism--each of which is described. The operation of all four mechanisms is governed and regulated by an executive program called the mechanism coordinator. As an example of the operation of the system, its behavior in finding solutions to increasingly difficult versions of the "Tower of Hanoi" puzzle is related. Suggestions are made as to appropriate directions for further research that would improve and broaden the capabilities of the system.

DESIGNING AN ARTIFICIAL STUDENT

Aiko Hormann
System Development Corporation
Santa Monica, California

To some of you, it might seem that this paper has a rather curious title. What do I mean by an artificial student? I intend the term to imply a non-human entity that is capable of exhibiting learning and problem-solving behavior--and the latter two terms, I confess, I am using on an intuitive basis. I am not trying to construct a mechanical man, or a robot; I am trying to develop a system of computer programs that will represent my conception of the inner workings of an artificial student. By means of programming, I wish to endow the computer with some of the capabilities and characteristics of learning and problem-solving behavior--behavior that, if observed in humans, we would call "intelligent."

Why do I want to construct an artificial student on a computer? In addition to the intrinsic interest in the questions of problem-solving and learning, there are some practical considerations and interesting speculations. Conventional uses of computers, even in a rather sophisticated man-machine partnership, seem to be based on 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. An attempt to shift this dividing line is a major concern of artificial intelligence research.

There are, of course, many technical and conceptual difficulties. The conceptual difficulties stem from our lack of real knowledge as to what happens--and how it happens--when we learn and behave intelligently. Technical difficulties exist because we have not completely mastered computer technology, either in programming or hardware design. (A subsidiary problem may be that man's ego is involved in such a shift; man tends to guard his sacred possession called "intelligence" against an invasion by nonhuman things.)

However, I believe that the role of the computer can be shifted from an idiot slave to a more-intelligent servant, and eventually to a colleague of the man. Man can then demand more and more responsibility and participation from machines in solving complex and difficult problems.

The goal of machine-as-colleague may be too distant to see clearly, but extending the present intellectual capacity of machines by means of an adaptive system is being attempted, and shows some progress.

Let us imagine a situation in which a man and a sufficiently developed adaptive system team up to attack a vaguely defined problem. The man can start with his partial knowledge of the problem situation, and can make some exploratory moves that express his initial assumptions about problem definitions and problem-solving methods. Clearer understanding of the problem situation can then be gained by trying out new ideas and methods and examining their consequences--always in close interaction with the system. In the particular problem context, new terms can be added to the system's vocabulary and new processes can be incorporated into its repertoire of skills. As the system gains experience, input information changes its meaning and gains new meanings. In addition, modifications and expansions of previously tried problem definitions and solution strategies can be made as the problem unfolds.

One way in which such an adaptive system might be constructed is the main topic of this paper. In designing the artificial student, many problem-solving techniques and behavior characteristics of humans have been incorporated, but the resulting system is not intended to be a model of human thought processes. In addition to asking what humans do when they think, we ask what kinds of mechanisms or operations are desirable in order to manifest the kind of behavior we want. I am interested in exploring a variety of possibilities with both human and nonhuman features and in discovering fruitful problem-solving and learning techniques or processes that are not necessarily a deliberate imitation of those used by humans. Therefore, my artificial student may accomplish its ends in ways different from human ways--when man invented a flying machine, it didn't have to flap its wings.

Now, how much capacity should the system be endowed with by preprogramming and how much should it be expected to acquire through learning? I do not want to start the artificial student at an "infant" level, nor is it feasible to prestore all the explicit responses required by all the conceivable problem situations. Is it possible to start the system at an intermediate level? It seems reasonable to make an initial molding by injecting some basic information-handling and decision-making capabilities, with specific emphasis on learning mechanisms. The learning mechanisms will then allow the artificial student to retain and profit from its own experience as I "train" the system on tasks of increasing complexity and variety.

This is the approach I have taken in designing the artificial student called Gaku. Gaku is a Japanese word denoting learning. Gaku's behavior is manifested as the combined effect of several factors--problem-solving mechanisms, learning mechanisms, memory organization, and specialized interpretations of such concepts as "abstraction," "generalization," "planning," and "induction." In Gaku, unlike most conventional methods of computation, prestored data and programs do not dictate <u>explicitly</u> how <u>exact</u> steps of a solution procedure are to be carried out for each new problem. Instead, each component of the system is given general rules of decision-making and information-handling operations. Gaku's behavior pattern in a specific problem context is then determined by these rules plus the results of its interaction with the external environment. In this sense, the information to prescribe Gaku's behavior is only implicitly contained in the initially prestored programs. Nor does such prestored information need to be complete, for control over Gaku's behavior is partly delegated to the external environment of the system.

The easiest way to give you a system description of Gaku is to sketch the working of its four mechanisms. Each is designed to work in a particular stage of problem-solving activity, but they all have one feature in common, namely a cycling process. It is abstracted from the behavior pattern generally observed or reported when humans meet a problem situation. When a man faces a problem with only partial knowledge about how to proceed, he usually starts with more-or-less random trial and error actions. Then he uses the results of his first attempt as a guide to his next try. Through such repetitions, his behavior becomes more selective, directed, and organized. He repeats the process until he either finds a solution or abandons the problem, and he often discovers his solution strategy in the course of the action itself.

I have attempted to use this iterative mode of operating in the basic design of Gaku. Other features and refinements in the system adapt the individual mechanisms to different purposes, but each mechanism depends ultimately on this fundamental cycling process. As we can see from Figure 1, the cycle passes through an analysis and test phase, a tentative selection or correction phase, and a consequence-generation phase. Based on the analysis and test phase, the system makes its first tentative selection of a solution procedure. The consequences of the selected course are generated, and these consequences are submitted to the analysis and test phase. There, they are compared with the description of the given task. In the light of the comparison, the task is reformulated and reanalyzed, and the system enters the selection phase for a second time. The newly selected actions or the modified actions result in a new set of consequences, and these in turn are analyzed and tested. The three phases are passed repeatedly until either a success or a failure is determined in the analysis and test phase.

The major components of the system are four mechanisms--a programming mechanism, a problem-oriented mechanism, a planning mechanism, and an induction mechanism. Each of the mechanisms uses the cycling process just described, but each works in a different stage of problem-solving activity. The "central office" for all this activity is a mechanism coordinator, which allocates the decision-making authorities and coordinates the functions of all the mechanisms. Figure 2 is a schematic representation of the relationships among these functional components. The mechanism coordinator at the top

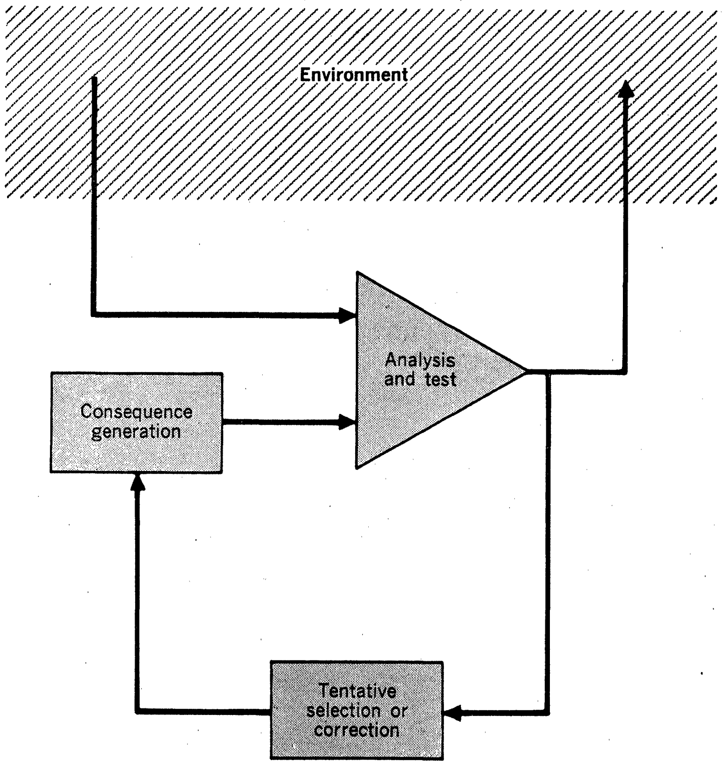

Figure 1. Three-Phase Cycle

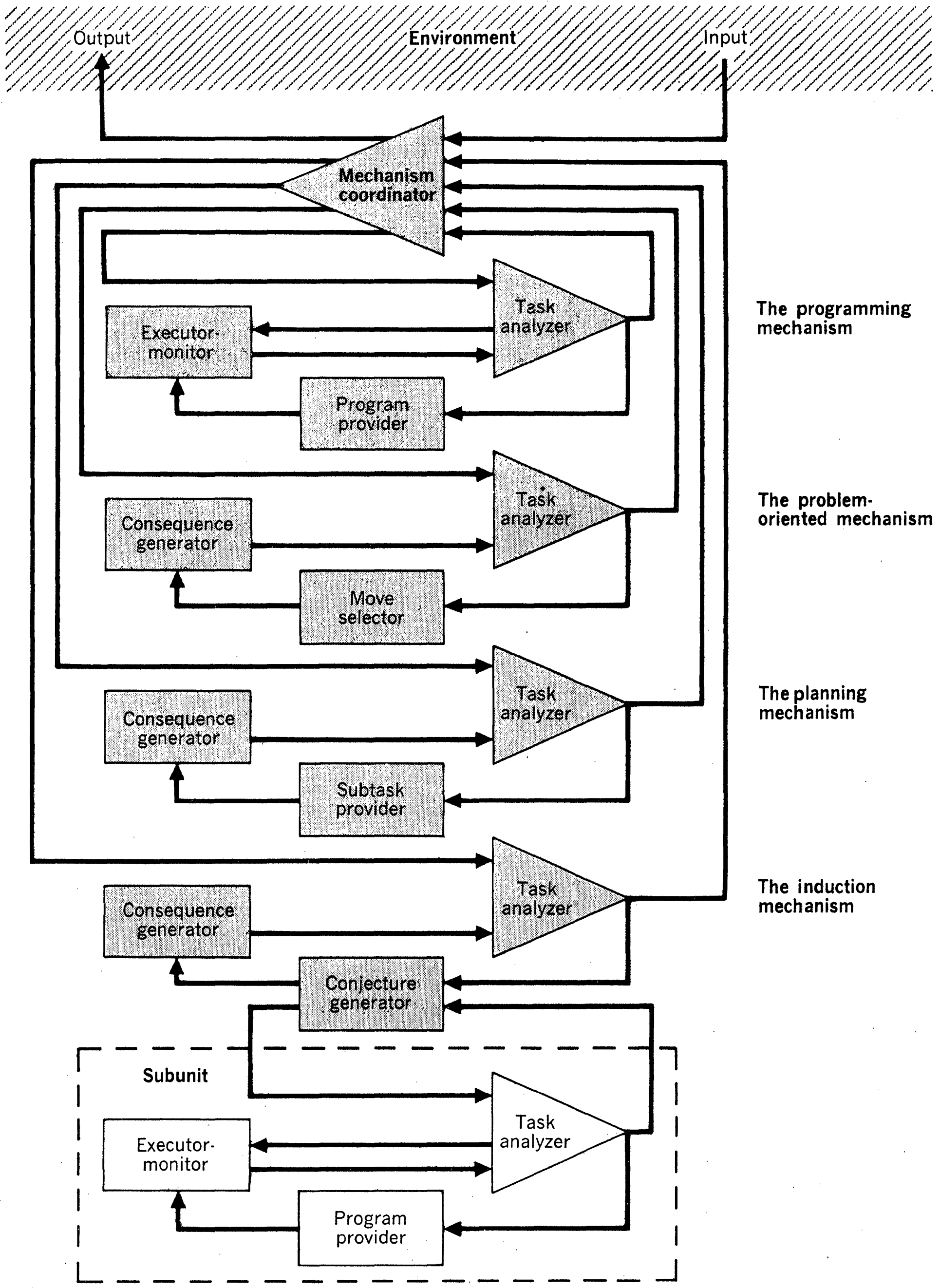

Figure 2. Four Mechanisms Coordinated

is the master program. In addition to its coordinating function, it takes care of the system's input and output channels. Through this coordinator, Gaku's interaction with its environment can be manifested.

The programming mechanism is responsible for internal programming; it generates programs and associated data internally from externally given descriptions or statements about what the programs do. Depending on the system's past experience and the particular task requested of the mechanism, it sometimes modifies previously generated subroutines, sometimes puts together available subroutines unmodified, or sometimes constructs new routines from basic operations. Usually it does a mixture of all three.

The problem-oriented mechanism generates and manipulates sequences of unit actions. These actions may be defined directly by the task environment, or they may be derived from the programming mechanism. It is the problem-oriented mechanism that constructs and actually carries out the required sequence of allowed moves that leads to the solution of a given problem--it is therefore the direct determiner of the behavior of the system. Some useful methods and techniques are provided for enabling the mechanism to search for a usable sequence of unit actions more efficiently than by exhaustive or random trial and error. These methods and techniques, however, are of a step-by-step nature, and can attack problems only in piecemeal fashion. For these reasons the problem-oriented mechanism tends to be confined within a narrow and restricted framework. For complex problems, therefore, the system needs a way to analyze problem structures and place guideposts on the road to the goal.

To this end, the system includes a planning mechanism. The planning mechanism takes a larger view, often at an abstract level, of a given task. After surveying the task as a whole, the planning mechanism subdivides the task into a hierarchy of subtasks, each of which is presumably easier to perform than the original task. This hierarchy of subtasks constitutes a rough sketch of a possible course of action that guides the problem-oriented mechanism.

The given task description usually indicates two states: the initial state (where you are now) and the goal state (where you want to be). The gap between the two states must be bridged somehow to go from the initial state to the goal state. Finding a complete solution can be thought of as establishing a valid path, one made up of legal unit actions or steps, between the two states. This is what the problem-oriented mechanism must eventually do. However, at an abstract level of planning, Gaku is not concerned with each unit action; instead, it activates the planning mechanism to handle wider-spaced steps or subtasks. These subtasks are defined by intermediate states or stepping stones, which lie within the gap between the two initially given states. These stepping stones constitute the plan of the solution and are then handled by the mechanism coordinator, which instructs the problem-oriented mechanism to solve one portion of the problem at a time.

To make efficient use of Gaku's past experience, the problem-oriented mechanism is also influenced by the induction mechanism, which takes a still larger view of a given task. The induction mechanism can survey the system's past experience with various problems and apply relevant experience to new problems. To call this "induction" is one of the instances in which I am indulging in an oversimplification. The mechanism really operates by a kind of mechanical pseudo-induction. In an article in Information and Control [Hormann, 1964] a fuller description of this mechanism is offered under the heading, "A Case of Mechanical Induction."

All these mechanisms represent a way of telling Gaku implicitly how to solve problems. You have probably heard the expression, "A computer can do only what it is told to do." That is true; but the word "only" implies a triviality that is far from realistic. There are many different ways in which a machine can be told what to do. In Gaku, the chosen way has not been to specify minute and exact steps of a solution procedure. Instead, each of the four mechanisms has been equipped with general rules for making decisions and with general information-handling capabilities. The objective is for the total system to be able to execute a discriminating search, through a large space of possible combinations, for those particular combinations that provide solutions to a given problem. Such a procedure is more effective than a random or exhaustive search. This way of specifying Gaku's behavior--through special-purpose mechanisms--is my approach to research on problem-solving by machine; it seeks an efficient way of dealing with problems whose solution requires search and experimentation along a course not clearly known in advance.

In order to evaluate Gaku's performance and detect its limitations, part of the system has been implemented on SDC's Philco-2000 computer. As a trial exercise, Gaku was given a sequence of increasingly difficult problems taken from the Tower of Hanoi puzzle. The upper half of Figure 3 shows the usual appearance of the puzzle, with three pegs and eight disks. The problem is to transfer the eight disks one at a time to either of the two empty pegs, never placing a larger disk on top of a smaller one. The lower half of the figure shows (schematically) the way in which the three-disk case of the puzzle was presented to Gaku. The letters "A," "B," and "C" represent the three pegs, and the circled numbers "1," "2," and "3" are the numbered disks, from the smallest to the largest. A slashed zero indicates an empty peg.

The three-disk case was the first problem given to Gaku. Lacking previous experience, Gaku exhibited a high proportion of trial-and-error behavior in this first exercise. Nevertheless, the over-all pattern of Gaku's behavior was not wholly exhaustive or random; this is because the problem-oriented mechanism activated special routines to analyze and characterize the problem in terms of the relative positions of the disks.

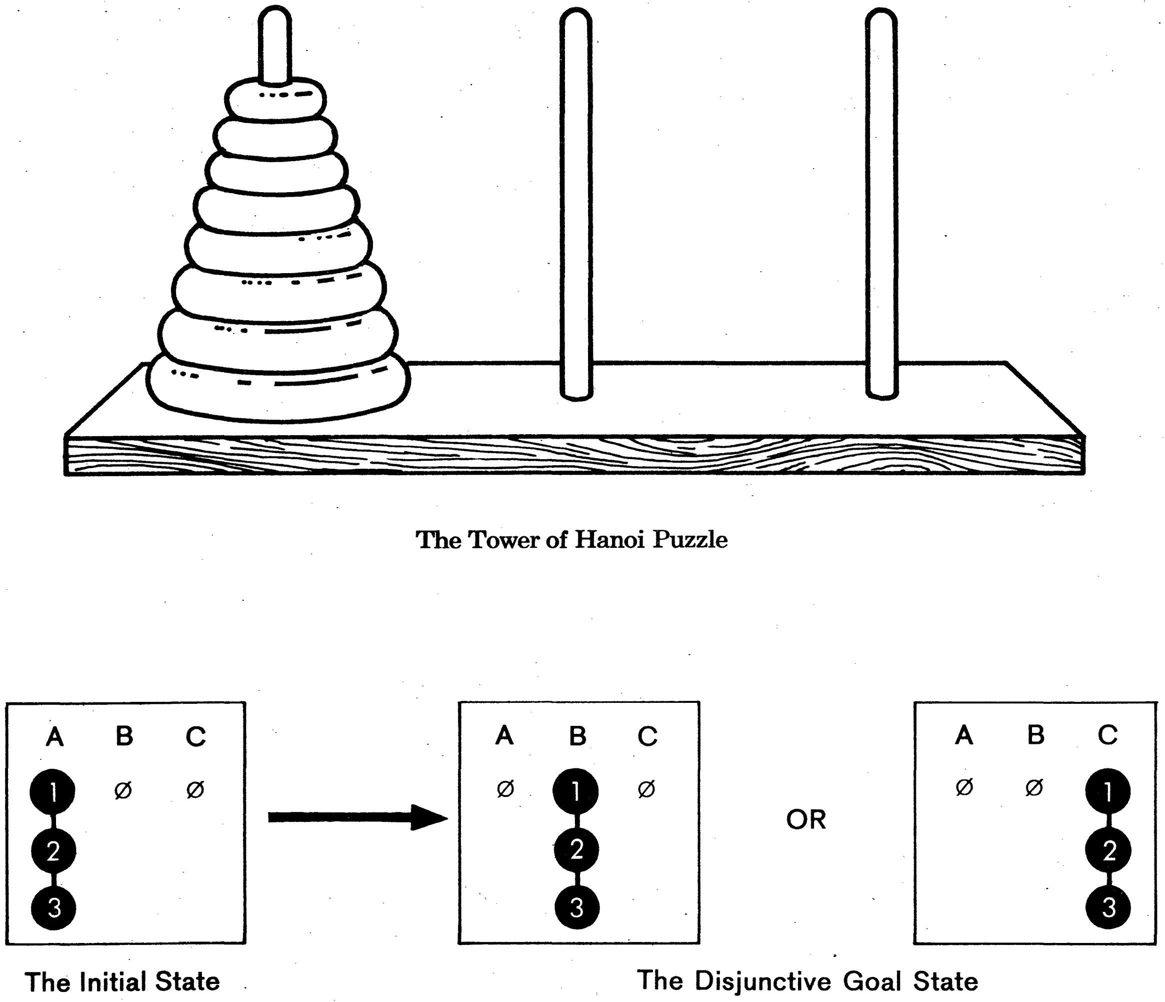

Figure 3. The Three-Disk Case of the Tower of Hanoi Puzzle

After a successful sequence of moves for the three-disk case had been found, Gaku was advanced to a slightly higher level of complexity--the four-disk case was presented. This time, Gaku attempted to use its past experience by activating the planning and induction mechanisms. The planning mechanism treated the four-disk problem as a combination of the three-disk problem and a one-disk problem. The subproblems thus created were attacked separately. The induction mechanism tried to find a pattern underlying the sequence of successful moves that lead to the goal. A conjectured pattern was tried out for the four-disk case as an extrapolation of the three-disk case. As it turned out, this extrapolation was only partially successful, so the conjectured pattern was modified by comparing its steps with the steps in the newly discovered successful solution.

This modified pattern was used at the next level--the five-disk case. Again, some discrepancies were found, but this time fewer than had been found before. In this manner, Gaku continued to attack progressively more difficult but similar problems, each time using its previous experience. Finally, through the induction mechanism, Gaku found a general solution pattern. (A detailed account of Gaku's performance in this trial has been published in the SDC Magazine in April, 1964.)

This experiment was successful in the sense that Gaku exhibited an aspect of learning and proved able to handle one small class of problems. However, in terms of its ability to handle a wide variety of complex tasks, Gaku's intellectual capacity is still far below the desired level. It is clear that there are many problems that must be solved and many investigations that must be conducted before even a modestly sophisticated learning system can be constructed.

In addition to the many refinements I must make in the internal structure of Gaku, a good communication means between Gaku and its tutor is badly needed. Such a communication means is imperative to an enhancement of Gaku's learning capacity. Research in the past has been mainly concerned with primary learning--that is, learning by firsthand experience in adapting to new situations. Secondary learning--learning from sources outside one's own experience--constitutes a large part of human learning, including learning to generalize. We acquire our generalization ability in classrooms and through textbooks. General principles are taught along with examples. Generalization processes, or acts of generalization, are demonstrated or reenacted. Gaku needs to be given a similar opportunity for secondary learning--an important kind of learning with which Gaku has not been provided so far. If secondary-learning capabilities were a part of the system, the human tutor could give Gaku exercise problems and hints of graded sequence on each subject area, as well as demonstrations of particular problem-solving techniques and generalization processes. The investigation of these possibilities might yield deeper insight into the problem of generalization and into the effects of interaction between primary and secondary learning.

I have described the rudimentary stages of an artificial student called Gaku. To circumscribe the area of investigation, I have made some strong assumptions about, and oversimplifications of, the very complex processes we call learning and problem-solving. For example, in talking about learning, I have purposely avoided considering fatigue, boredom, motivation--all the emotional factors that subtly but strongly influence human performance. My approach is similar to that employed in the natural sciences--first investigating deliberately simplified cases, considering only a few variables, restricting the behavior of these variables to simple, known functions, then introducing more complex cases, studying the effects of changes, repeating processes, and continually attempting to approach more realistic situations. Following this pattern, I have alternated the phases of design and evaluation in Gaku's development, trying to discern the system's shortcomings and find new ways to make it more efficient and more general. As my research progresses, I hopeto be able to report further development toward the transition I spoke of earlier--the transition of machines from rigidly obedient servants to cooperating colleagues.

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.

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.

Minsky, M. Steps toward artificial intelligence. _Proc. IRE_, 1961, _49_, 8-30.

Minsky, M. Descriptive languages and problem solving. _Proc. WJCC_, 1961, _19_, 215-218.

Newell, A., Shaw, J. C., and Simon, H. A. A general problem-solving program for a computer. _Computers and Automation_, 1959, _8_, 10-17.

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.

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.

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.

SDC