It will be seen that the effect of expressing approval when a particular figure has been printed is to increase the chance of that figure being printed on subsequent occasions, and so, if done sufficient times, to give the appearance of a conditioned reflex action having been established.

### Generalized Learning Programs

The construction of a learning program of the above type presents no difficulty to anyone who is familiar with the technique of programming. However, such a program makes it possible to teach the machine only those things which the programmer had in mind when he wrote the program. For example, the program just described would not enable the operator to teach the machine to print different figures after alternate stimuli. If the programmer had wished to do so, he could have allowed for this possibility in his program. In fact, at the expense of making the program longer and more complicated, he could include any number of extra features, but obviously he could not think of every possible experiment which anyone might wish to try on the machine.

All the programmer is doing, in fact, is to program the action of the machine as it were at one remove; when he writes the program he visualizes, if not in complete detail, at any rate in general terms, all the possible actions which the machine can take in response to legitimate actions on the part of the operator.

Such programms are not, therefore, as interesting as at first sight might appear from the point of view of this article. What is wanted is a "generalized" learning program, which would enable an operator to teach the machine anything he chose, whether the idea of his doing so had occurred to the programmer or not. I believe that such a program would not be a mere elaboration of the simple learning programs which have been constructed up to date but would need to be based on some entirely new ideas. Presumably the program would modify and extend itself as the learning process went on.

As I have pointed out, existing machines contain the means for this extension; the difficulty is to construct a program to make use of them. If such a program could be made then it would be possible to teach the machine in much the same way as a child is taught.

Whether the new ideas I have referred to will be forthcoming, it is hard to say. Certainly, for the present, progress appears to be held up. Perhaps this will give comfort to those who cannot bear the idea of machines thinking; on the other hand it may stimulate others to further effort.

### Bibliography

E. C. Berkeley, "Giant Brains," John Wiley & Sons, New York, N. Y.; 1949.
A. G. Oettinger, "Programming a digital computer to learn," *Phil. Mag.*, vol. 7, no. 43, pp. 1243–1263; 1952.
C. E. Shannon, "Programming a computer for playing chess," *Ibid.*, vol. 7, no. 41, pp. 256–275; 1950.
A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460; 1950.
M. V. Wilkes, "Can machines think?" *The Spectator*, no. 6424, pp. 177–178; 1951. "Automatic calculating machines," *Jour. Roy. Soc. A.*, vol. 100, pp. 56–90; 1951.
M. V. Wilkes, D. J. Wheeler, and S. Gill, "The Preparation of Programs for an Electronic Digital Computer," Addison-Wesley Cambridge, Mass.; 1951.

# Computers and Automata*

CLAUDE E. SHANNON†, FELLOW, IRE

C. E. Shannon first became known for a paper in which he applied Boolean Algebra to relay switching circuits; this laid the foundation for the present extensive application of Boolean Algebra to computer design. Dr. Shannon, who is engaged in mathematical research at Bell Telephone Laboratories, is an authority on information theory. More recently he received wide notice for his ingenious maze-solving mechanical mouse, and he is well-known as one of the leading explorers into the exciting, but uncharted world of new ideas in the computer field.

The Editors asked Dr. Shannon to write a paper describing current experiments, and speculations concerning future developments in computer logic. Here is a real challenge for those in search of a field where creative ability, imagination, and curiosity will undoubtedly lead to major advances in human knowledge.—*The Editor*

***Summary*—This paper reviews briefly some of the recent developments in the field of automata and nonnumerical computation. A number of typical machines are described, including logic machines, game-playing machines and learning machines. Some theoretical questions and developments are discussed, such as a comparison of computers and the brain, Turing's formulation of computing machines and von Neumann's models of self-reproducing machines.**

* Decimal classification: 621.385.2. Original manuscript received by the Institute, July 17, 1953.
† Bell Telephone Laboratories, Murray Hill, N. J.

### Introduction

SAMUEL BUTLER, in 1871, completed the manuscript of a most engaging social satire, *Erewhon*. Three chapters of *Erewhon*, originally appearing under the title "Darwin Among the Machines," are a witty parody of *The Origin of Species*. In the topsy-turvy logic of satirical writing, Butler sees machines as gradually evolving into higher forms. He considers the classification of machines into genera, species and vari-

eties, their feeding habits, their rudimentary sense organs, their reproductive and evolutionary mechanisms (inefficient machines force men to design more efficient ones), tendencies toward reversion, vestigial organs, and even the problem of free will in machines.

Rereading *Erewhon* today one finds "The Book of the Machines" disturbingly prophetic. Current and projected computers and control systems are indeed assuming more and more the capacities and functions of animals and man, to a far greater degree, in fact, than was envisaged by Butler.

The bread-and-butter work of large-scale computers has been the solution of involved numerical problems. To many of us, however, the most exciting potentialities of computers lie in their ability to perform non-numerical operations—to work with logic, translate languages, design circuits, play games, co-ordinate sensory and manipulative devices and, generally, assume complicated functions associated with the human brain.

Non-numerical computation is by no means an unproven offspring of the more publicized arithmetic calculation. The shoe is rather on the other foot. A hundred years ago Charles Babbage was inspired in the design of his remarkably prescient analytical engine by a portrait woven in silk on a card controlled Jacquard loom—a device then in existence half a century. The largest and most reliable current information processing machine is still the automatic telephone system. Our factories are filled with ingenious and unsung devices performing almost incredible feats of sensing, processing and transporting materials in all shapes and forms. Railway and power systems have elaborate control and protective networks against accidents and human errors.

These, however, are all special-purpose automata. A significant new concept in non-numerical computation is the idea of a general-purpose programmed computer—a device capable of carrying out a long sequence of elementary orders analogous to those of a numerical computer. The elementary orders, however, will relate not to operations on numbers but to physical motions, operations with words, equations, incoming sensory data, or almost any physical or conceptual entities.

This paper reviews briefly some of the research in non-numerical computation and discusses certain of the problems involved. The field is currently very active and in a short paper only a few sample developments can be mentioned.

## The Brain and Computers

The brain has often been compared, perhaps overenthusiastically, with computing machines. It contains roughly $10^{10}$ active elements called neurons. Because of the all or none law of nervous action, neurons bear some functional resemblance to our binary computer elements, relays, vacuum tubes or transistors. The number of elements is six orders of magnitude greater than our largest computers. McCullough has picturesquely put it that a computer with as many tubes as a man has neurons would require the Empire State building to house it, Niagara Falls to power it and the Niagara river to cool it. The use of transistors in such a comparison would improve the figures considerably, power requirements coming down to the hundreds of kilowatt range (the brain dissipates some 25 watts) and size requirements (with close packing) comparable to an ordinary dwelling. It may also be argued that the increased speed of electronic components by a factor of, say, $10^3$ might be partially exchangeable against equipment requirements.

Comparisons of this sort should be taken well salted—our understanding of brain functioning is still, in spite of a great deal of important and illuminating research, very primitive. Whether, for example, the neuron itself is the proper level for a functional analysis is still an open question. The random structure at the neural level in number, placement and interconnections of the neurons, suggests that only the statistics are important at this stage, and, consequently, that one might average over local structure and functioning before constructing a mathematical model.

The similarities between the brain and computers have often been pointed out. The differences are perhaps more illuminating, for they may suggest the important features missing from our best current brain models. Among the most important of these are:

1. Differences in size. Six orders of magnitude in the number of components takes us so far from our ordinary experience as to make extrapolation of function next to meaningless.
2. Differences in structural organization. The apparently random local structure of nerve networks is vastly different from the precise wiring of artificial automata, where a single wrong connection may cause malfunctioning. The brain somehow is designed so that overall functioning does not depend on the exact structure in the small.
3. Differences in reliability organization. The brain can operate reliably for decades without really serious malfunctioning (comparable to the meaningless gibberish produced by a computer in trouble conditions) even though the components are probably individually no more reliable than those used in computers.
4. Differences in logical organization. The differences here seem so great as to defy enumeration. The brain is largely self-organizing. It can adapt to an enormous variety of situations tolerably well. It has remarkable memory classification and access features, the ability to rapidly locate stored data via numerous "coordinate systems." It can set up stable servo systems involving complex relations between its sensory inputs and motor outputs, with great facility. In contrast, our digital computers look like idiot savants. For long chains of arithmetic operations a digital computer runs circles around the best humans. When we try to program computers for other activities their entire organization seems clumsy and inappropriate.

5. Differences in input-output equipment. The brain is equipped with beautifully designed input organs, particularly the ear and the eye, for sensing the state of its environment. Our best artificial counterparts, such as Shepard's Analyzing Reader for recognizing and transcribing type, and the "Audrey" speech recognition system which can recognize the speech sounds for the ten digits seem pathetic by comparison. On the output end, the brain controls hundreds of muscles and glands. The two arms and hands have some sixty independent degrees of freedom. Compare this with the manipulative ability of the digitally controlled milling machine developed at M.I.T., which can move its work in but three co-ordinates. Most of our computers, indeed, have no significant sensory or manipulative contact with the real world but operate only in an abstract environment of numbers and operations on numbers.

## Turing Machines

The basic mathematical theory of digital computers was developed by A. M. Turing in 1936 in a classic paper "On Computable Numbers with an Application to the Entscheidungsproblem." He defined a class of computing machines, now called Turing machines, consisting basically of an infinite paper tape and a computing element. The computing element has a finite number of internal states and is capable of reading from and writing on one cell of the tape and of moving it one cell to the right or left. At a given time, the computing element will be in a certain state and reading what is written in a particular cell of the tape. The next operation will be determined by the current state and the symbol being read. This operation will consist of assuming a new state and either writing a new symbol (in place of the one currently read) or moving to the right or to the left. It is possible for machines of this type to compute numbers by setting up a suitable code for interpreting the symbols. For example, in Turing's formulation the machines print final answers in binary notation on alternate cells of the tape, using the other cells for intermediate calculations.

It can be shown that such machines form an extremely broad class of computers. All ordinary digital computers which do not contain a random or probabilistic element are equivalent to some Turing machine. Any number that can be computed on these machines, or in fact by any ordinary computing process, can be computed by a suitable Turing machine. There are, however, as Turing showed, certain problems that cannot be solved and certain numbers that cannot be computed by any Turing machine. For example, it is not possible to construct a Turing machine which, given a suitably coded description of another Turing machine, can always tell whether or not the second Turing machine will continue indefinitely to print symbols in the squares corresponding to the final answer. It may, at a certain point in the calculation, relapse into an infinite intermediate computation. The existence of mechanically unsolvable problems of this sort is of great interest to logicians.

Turing also developed the interesting concept of a universal Turing machine. This is a machine with the property that if a suitably coded description of any Turing machine is printed on its tape, and the machine started at a suitable point and in a suitable state, it will then act like the machine described, that is, compute (normally at a much slower rate) the same number that the described machine would compute. Turing showed that such universal machines can be designed. They of course are capable of computing any computable number. Most digital computers, provided they have access to an unlimited memory of some sort, are equivalent to universal Turing machines and can, in principle, imitate any other computing machine and compute any computable number.

The work of Turing has been generalized and reformulated in various ways. One interesting generalization is the notion of A computability. This relates to a class of Turing type machines which have the further feature that they can, at certain points of the calculation, ask questions of a second "oracular" device, and use the answers in further calculations. The oracular machine may for example have answers to some of the unsolvable problems of ordinary Turing machines, and consequently enable the solution of a larger class of problems.

## Logic Machines

Boolean algebra can be used as a mathematical tool for studying the properties of relay and switching circuits. Conversely, it is possible to solve problems of Boolean algebra and formal logic by means of simple relay circuits. This possibility has been exploited in a number of logic machines. A typical machine of this kind, described by McCallum and Smith, can handle logical relations involving up to seven classes or truth variables. The required relations among these variables, given by the logical problem at hand, are plugged into the machine by means of a number of "connective boxes." These connective boxes are of six types and provide for the logical connectives "not," "and," "or," "or else," "if and only if," and "if-then." When the connections are complete, starting the machine causes it to hunt through the $2^7 = 128$ combinations of the basic variables, stopping at all combinations which satisfy the constraints. The machine also indicates the number of "true" variables in each of these states. McCallum and Smith give the following typical problem that may be solved on the machine:

*It is known that salesmen always tell the truth and engineers always tell lies. G and E are salesmen. C states that D is an engineer. A declares that B affirms that C asserts that D says that E insists that F denies that G is a salesman. If A is an engineer, how many engineers are there?*

A very suggestive feature in this machine is a selec-

tive feedback system for hunting for particular solutions of the logical equations without an exhaustive search through all possible combinations. This is achieved by elements which sense whether or not a particular logical relation is satisfied. If not, the truth variables involved in this relation are caused to oscillate between their two possible values. Thus, variables appearing in unsatisfied relations are continually changing, while those appearing only in satisfied relations do not change. If ever all relations are simultaneously satisfied the machine stops at that particular solution. Changing only the variables in unsatisfied relations tends, in a general way, to lead to a solution more rapidly than methodical exhaustion of all cases, but, as is usually the case when feedback is introduced, leads to the possibility of continual oscillation. McCallum and Smith point out the desirability of making the changes of the variables due to the feedback unbalance as random as possible, to enable the machine to escape from periodic paths through various states of the relays.

## Game Playing Machines

The problem of designing game-playing machines is fascinating and has received a good deal of attention. The rules of a game provide a sharply limited environment in which a machine may operate, with a clearly defined goal for its activities. The discrete nature of most games matches well the digital computing techniques available without the cumbersome analog-digital conversion necessary in translating our physical environment in the case of manipulating and sensing machines.

Game playing machines may be roughly classified into types in order of increasing sophistication:

1. Dictionary-type machines. Here the proper move of the machine is decided in advance for each possible situation that may arise in the game and listed in a "dictionary" or function table. When a particular position arises, the machine merely looks up the move in the dictionary. Because of the extravagant memory requirements, this rather uninteresting method is only feasible for exceptionally simple games, e.g., tic-tac-toe.
2. Machines using rigorously correct playing formulas. In some games, such as Nim, a complete mathematical theory is known, whereby it is possible to compute by a relatively simple formula, in any position that can be won, a suitable winning move. A mechanization of this formula provides a perfect game player for such games.
3. Machines applying general principles of approximate validity. In most games of interest to humans, no simple exact solution is known, but there are various general principles of play which hold in the majority of positions. This is true of such games as checkers, chess, bridge, poker and the like. Machines may be designed applying such general principles to the position at hand. Since the principles are not infallible, neither are the machines, as indeed, neither are humans.
4. Learning machines. Here the machine is given only the rules of the game and perhaps an elementary strategy of play, together with some method of improving this strategy through experience. Among the many methods that have been suggested for incorporation of learning we have:

   a) trial-and-error with retention of successful and elimination of unsuccessful possibilities;

   b) imitation of a more successful opponent;

   c) "teaching" by approval or disapproval, or by informing the machine of the nature of its mistakes; and finally

   d) self-analysis by the machine of its mistakes in an attempt to devise general principles.

Many examples of the first two types have been constructed and a few of the third. The fourth type, learning game-players, is reminiscent of Mark Twain's comment on the weather. Here is a real challenge for the programmer and machine designer.

Two examples of the third category, machines applying general principles, may be of interest. The first of these is a machine designed by E. F. Moore and the writer for playing a commercial board game known as Hex. This game is played on a board laid out in a regular hexagon pattern, the two players alternately placing black and white pieces in unoccupied hexagons. The entire board forms a rhombus and Black's goal is to connect the top and bottom of this rhombus with a continuous chain of black pieces. White's goal is to connect the two sides of the rhombus with a chain of white pieces. After a study of this game, it was conjectured that a reasonably good move could be made by the following process. A two-dimensional potential field is set up corresponding to the playing board, with white pieces as positive charges and black pieces as negative charges. The top and bottom of the board are negative and the two sides positive. The move to be made corresponds to a certain specified saddle point in this field.

To test this strategy, an analog device was constructed, consisting of a resistance network and gadgetry to locate the saddle points. The general principle, with some improvements suggested by experience, proved to be reasonably sound. With first move, the machine won about seventy per cent of its games against human opponents. It frequently surprised its designers by choosing odd-looking moves which, on analysis, proved sound. We normally think of computers as expert at long involved calculations and poor in generalized value judgments. Paradoxically, the positional judgment of this machine was good; its chief weakness was in end-game combinatorial play. It is also curious that the Hex-player reversed the usual computing procedure in that it solved a basically digital problem by an anlog machine.

The game of checkers has recently been programmed into a general-purpose computer, using a "general principle" approach. C. S. Strachey used a method similar to

one proposed by the writer for programming chess—an investigation of the possible variations for a few moves and a minimax evaluation applied to the resulting positions. The following is a sample game played by the checker program with notes by Strachey. (The white squares are numbered consecutively, 0–31, from left to right and top to bottom. Numbers in parentheses indicate captures.)

| MACHINE | STRACHEY |
|---|---|
| 11—15 | 23—18 |
| 7—11 | 21—17 |
| 8—12 | 20—16 *a* |
| 12—21 (16) | 25—16 (21) |
| 9—14 ! *b* | 18— 9 (14) |
| 6—20 (16, 9) *c* | 27—23 |
| 2— 7 *d* | 23—18 |
| 5— 8 | 18—14 |
| 8—13 *e* | 17— 8 (13) |
| 4—13 (8) | 14— 9 |
| 1— 5 *f* | 9— 6 |
| 15—19 | 6— 1 (K) |
| 5— 9 | 1— 6 ?*g* |
| 0— 5 ! *h* | 6—15 (10) |
| 11—25 (22, 15) | 30—21 (25) |
| 13—17 | 21—14 (17) |
| 9—18 (14) | 24—21 |
| 18—23 | 26—22 |
| 23—27 | 22—17 |
| 5— 8 *i* | 17—14 |
| 8—13 | 14— 9 |
| 19—23 | 9— 6 |
| 23—26 *j* | 31—22 (26) |
| 27—31 (K) | 6— 2 (K) |
| 7—10 | 2— 7 |
| 10—15 | 21—16 ?*k* |
| 3—10 (7) | 16— 9 (13) |
| 10—14 | 9— 6 |
| 15—19 | 6— 2 (K) |
| 31—27 *m* | 2— 6 |
| 27—31 *m* | 6—10 |
| 31—26 *n* | 10—17 (14) |
| 19—23 | 29—25 |
| 26—31 *p* | |

Notes:

*a*) An experiment on my part—the only deliberate offer I made. I thought, wrongly, that it was quite safe.
*b*) Not foreseen by me.
*c*) Better than 5–21 (9, 17).
*d*) A random move (zero value). Shows the lack of a constructive plan.
*e*) Another random move of zero value. Actually rather good.
*f*) Bad. Ultimately allows me to make a King. 10–14 would would have been better.
*g*) A bad slip on my part.
*h*) Taking full advantage of my slip.
*i*) Bad, unblocks the way to a King.
*j*) Sacrifice in order to get a King (not to stop me Kinging). A good move, but not possible before 19–23 had been made by chance.
*k*) Another bad slip on my part.
*m*) Purposeless. The strategy is failing badly in the end game.
*n*) Too late.
*p*) Futile. The game was stopped at this point as the outcome was obvious.

While obviously no world champion, the machine is certainly better than many humans. Strachey points out various weaknesses in the program, particularly in certain end-game positions, and suggests possible improvements.

## Learning Machines

The concept of learning, like those of thinking, consciousness and other psychological terms, is difficult to define precisely in a way acceptable to the various interested parties. A rough formulation might be framed somewhat as follows. Suppose that an organism or a machine can be placed in, or connected to, a class of environments, and that there is a measure of "success" or "adaptation" to the environment. Suppose further that this measure is comparatively local in time, that is, that one can measure the success over periods of time short compared to the life of the organism. If this local measure of success tends to improve with the passage of time, for the class of environments in question, we may say that the organism or machine is learning to adapt to these environments relative to the measure of success chosen. Learning achieves a quantitative significance in terms of the broadness and complexity of the class of environments to which the machine can adapt. A chess playing machine whose frequency of wins increases during its operating life may be said by this definition to be learning chess, the class of environments being the chess players who oppose it, and the adaptation measure, the winning of games.

A number of attempts have been made to construct simple learning machines. The writer constructed a maze-solving device in which an arbitrary maze can be set up in a five-by-five array of squares, by placing partitions as desired between adjacent squares. A permanently magnetized "mouse," placed in the maze, blunders about by a trial and error procedure, striking various partitions and entering blind alleys until it eventually finds its way to the "food box." Placed in the maze a second time, it will move directly to the food box from any part of the maze that it has visited in its first exploration, without errors or false moves. Placed in other parts of the maze, it will blunder about until it reaches a previously explored part and from there go directly to the goal. Meanwhile it will have added the information about this part of the maze to its memory, and if placed at the same point again will go directly to the goal. Thus by placing it in the various unexplored parts of the maze, it eventually builds up a complete pattern of information and is able to reach the goal directly from any point.

If the maze is now changed, the mouse first tries the old path, but on striking a partition starts trying other directions and revising its memory until it eventually reaches the goal by some other path. Thus it is able to forget an old solution when the problem is changed.

The mouse is actually driven by an electromagnet moving beneath the maze. The motion of the electromagnet is controlled by a relay circuit containing about 110 relays, organized into a memory and a computing circuit, somewhat after that of a digital computer.

The maze-solver may be said to exhibit at a very primitive level the abilities to (1) solve problems by trial and error, (2) repeat the solutions without the errors, (3) add and correlate new information to a partial solution, (4) forget a solution when it is no longer applicable.

Another approach to mechanized learning is that of suitably programming a large-scale computer. A. E. Oettinger has developed two learning programs for the Edsac computer in Cambridge, England. In the first of these, the machine was divided into two parts, one part playing the role of a learning machine and the second its environment. The environment represented abstractly a number of stores in which various items might be purchased, different stores stocking different classes of items. The learning machine faced the problem of learning where various items might be purchased. Starting off with no previous knowledge and a particular item to be obtained, it would search at random among the stores until the item was located. When finally successful, it noted in its memory where the article was found. Sent again for the same article it will go directly to the shop where it previously obtained this article. A further feature of the program was the introduction of a bit of "curiosity" in the learning machine. When it succeeded in finding article number $j$ in a particular shop it also noticed whether or not that shop carried articles $j-1$ and $j+1$ and recorded these facts in its memory.

The second learning program described by Oettinger is modeled more closely on the conditioned reflex behavior of animals. A stimulus of variable intensity can be applied to the machine in the form of an input integer. To this stimulus the machine may respond in a number of different ways indicated by an output integer. After the response, it is possible for the operator to indicate approval or disapproval by introducing a third integer at a suitable point. When the machine starts operating, its responses to stimuli are chosen at random. Indication of approval improves the chances for the response immediately preceding; indication of disapproval reduces this chance. Furthermore, as a particular response is learned by conditioning it with approval, the stimulus required for this response decreases. Finally, there is a regular decay of thresholds when no approval follows a response.

Further embellishments of programs of this sort are limited only by the capacity of the computer and the energy and ingenuity of the program designer. Unfortunely, the elementary orders available in most large-scale computers are poorly adapted to the logical requirements of learning programs, and the machines are therefore used rather inefficiently. It may take a dozen or more orders to represent a logically simple and frequently used operation occurring in a learning routine.

Another type of learning machine has been constructed by D. W. Hagelbarger. This is a machine designed to play the game of matching pennies against a human opponent. On the front panel of the machine are a start button, two lights marked + and −, and a key switch whose extreme positions are also marked + and −. To play against the machine, the player chooses + or −, and then pushes the start button. The machine will then light up one of the two lights. If the machine matches the player, that is, lights the light corresponding to the choice of the player, the machine wins; otherwise the player wins. When the play is complete, the player registers by appropriate movement of the key switch the choice he made.

The machine is so constructed as to analyze certain patterns in the players' sequence of choices, and attempt to capitalize on these patterns when it finds them. For example, some players have a tendency if they have won a round, played the same thing and won again, to then change their choice. The machine keeps count of these situations and, if such tendencies appear, plays in such a way as to win. When such patterns do not appear the machine plays at random.

It has been found the machine wins about 55–60 per cent of the rounds, while by chance or against an opponent that played strictly at random it would win only 50 per cent of the time. It appears to be quite difficult for a human being to produce a random sequence of pluses and minuses (to insure the 50 per cent wins he is entitled to by the theory of games) and even more difficult to actually beat the machine by leading it on to suspect patterns, and then reversing the patterns.

A second penny-matching machine was designed by the writer, following the same general strategy but using a different criterion to decide when to play at random and when to assume that an apparent behavior pattern is significant. After considerable discussion as to which of these two machines could beat the other, and fruitless attempts to solve mathematically the very complicated statistical problem involved when they are connected together, the problem was relegated to experiment. A third small machine was constructed to act as umpire and pass the information back and forth between the machines concerning their readiness to make a move and the choices made. The three machines were then plugged together and allowed to run for a few hours, to the accompaniment of small side-bets and loud cheering. Ironically, it turned out that the smaller, more precipitate of the two machines consistently beat the larger, more deliberate one in a ratio of about 55 to 45.

A still different type of learning machine was devised by W. Ross Ashby who christened it the Homeostat. Homeostasis, a word coined by Walter B. Cannon, relates to an animal's ability to stabilize, by feedback, such biological variables as body temperature, chemical concentrations in the blood stream, etc. Ashby's device is a kind of self-stabilizing servo system. The first model of the Homeostat contained four interconnected servos. The cross-connections of these servos passed through four stepping switches and resistors connected to the points of the steppers. Thus the effect of unbalance in the other three loops on a particular loop depended on the values of the resistors being contacted by the stepper associated with that loop. When any one of the servos was sufficiently out of balance, a corresponding limit relay would operate and cause the corresponding stepping switch to advance one point. Now normally, a

servo system with four degrees of freedom and random cross- and self-gain figures will not be stable. If this occurred, one or more of the stepping switches would advance and a new set of resistors would produce a new set of gain figures. If this set again proved unstable, a further advance of the steppers would occur until a stable situation was found. The values of the resistors connected to the stepping switches were chosen by random means (using a table of randon numbers). Facilities were provided for introducing many arbitrary changes or constraints among the servos. For example, their connections could be reversed, two of them could be tied together, one of them held at a fixed value, etc. Under all these conditions, the mechanism was able to find a suitable stable position with all the servos in balance. Considering the machine's goal to be that of stabilizing the servos, and the environment to be represented by the various alterations and constraints introduced by the operator, the Homeostat may be said to adapt to its environment.

Certain features of the Homeostat are quite attractive as a basis for learning machines and brain models. It seems in certain ways to do a bit more than was explicitly designed into it. For example, it has been able to stabilize under situations not anticipated when the machine was constructed. The use of randomly chosen resistors is particularly suggestive and reminiscent of the random connections among neurons in the brain. Ashby, in fact, believes that the general principle embodied in the Homeostat, which he calls ultra-stability, may underlie the operation of the animal nervous system. One of the difficulties of a too direct application of this theory is that, as Ashby points out, the time required for finding a stable solution grows more or less exponentially with the number of degrees of freedom. With only about 20 degrees of freedom, it would require many lifetimes to stabilize one system. Attempts to overcome this difficulty lead to rather involved conceptual constructions, so involved that it is extremely difficult to decide just how effectively they would operate. Our mathematical tools do not seem sufficiently sharp to solve these problems and further experimental work would be highly desirable.

## Self-Reproducing Machines

In *Erewhon* the reproduction process in machines was pictured as a kind of symbiotic co-operation between man and machines, the machines using man as an intermediary to produce new machines when the older ones were worn out. Man's part is akin to that of the bee in the fertilization of flowers. Recently von Neumann has studied at an abstract level the problem of true self-reproduction in machines, and has formulated two different mathematical models of such "machines."

The first of these may be pictured somewhat as follows. "Machines" in the model are constructed from a small number (of the order of twenty) types of elementary components. These components have relatively simple functions, for example, girders for structural purposes, elementary logical elements similar to simplified relays or neurons for computing, sensing components for detecting the presence of other elements, joining components (analogous to a soldering iron) for fastening elements together, and so on. From these elements, various types of machines may be "constructed." In particular, it is possible to design a kind of universal construction machine, analogous to Turing's universal computing machine. The universal constructing machine can be fed a sequence of instructions, similar to the program of a digital computer, which describe in a suitable code how to construct any other machine that can be built with the elementary components. The universal constructing machine will then proceed to hunt for the needed components in its environment and build the machine described on its tape. If the instructions to the universal constructing machine are a description of the universal constructing machine itself, it will proceed to build a copy of itself, and would be a self-reproducing machine except for the fact that the copy is not yet supplied with a set of instructions. By adding to the universal machine what amounts to a tape-copying device and a relatively simple controlling device, a true self-reproducing machine is obtained. The instructions now describe the original universal machine with the addition of the tape reproducer and the controlling device. The first operation of the machine is to reproduce this entity. The controlling device then sends the instruction tape through the tape reproducer to obtain a copy, and places this copy in the second machine. Finally, it turns the second machine on, which starts reading its instructions and building a third copy, and so ad infinitum.

More recently, von Neumann has turned from this somewhat mechanical model to a more abstract self-reproducing structure—one based on a two-dimensional array of elementary "cells." Each cell is of relatively simple internal structure, having, in fact, something like thirty possible internal states, and each cell communicates directly only with its four neighbors. The state of a cell at the next (quantized) step in time depends only on the current state of the cell and the states of its four neighbors. By a suitable choice of these state transitions it is possible to set up a system yielding a kind of self-reproducing structure. A group of contiguous cells can act as an organic unit and operate on nearby quiescent cells in such a way as to organize a group of them into an identical unit.

This second model avoids many of the somewhat extraneous problems of locating, recognizing and positioning components that were inherent in the first model, and consequently leads to a simpler mathematical formulation. Furthermore, it has certain analogies with various chemical and biological problems, such as those of crystal and gene reproduction, while the first model is more closely related to the problem of large scale animal reproduction.

An interesting concept arising from both models is the notion of a critical complexity required for self-reproduction. In either case, only sufficiently complicated "machines" will be capable of self-reproduction. Von Neumann estimates the order of tens of thousands of components or cells to obtain this property. Less complicated structures can only construct simpler "machines" than themselves, while more complicated ones may be capable of a kind of evolutionary improvement leading to still more complicated organisms.

## Challenge to the Reader

We hope that the foregoing sample of non-numerical computers may have stimulated the reader's appetite for research in this field. The problem of how the brain works and how machines may be designed to simulate its activity is surely one of the most important and difficult facing current science. Innumerable questions demand clarification, ranging from experimental and development work on the one hand to purely mathematical research on the other. Can we design significant machines where the connections are locally random? Can we organize machines into a hierarchy of levels, as the brain appears to be organized, with the learning of the machine gradually progressing up through the hierarchy? Can we program a digital computer so that (eventually) 99 per cent of the orders it follows are written by the computer itself, rather than the few per cent in current programs? Can a self-repairing machine be built that will locate and repair faults in its own components (including components in the maintenance part of the machine)? What does a random element add in generality to a Turing machine? Can manipulative and sensory devices functionally comparable to the hand and eye be developed and coordinated with computers? Can either of von Neumann's self-reproducing models be translated into hardware? Can more satisfactory theories of learning be formulated? Can a machine be constructed which will design other machines, given only their broad functional characteristics? What is a really good set of orders in a digital computer for general purpose non-numerical computation? How can a computer memory be organized to learn and remember by association, in a manner similar to the human brain?

We suggest these typical questions, and the entire automata field, as a challenge to the reader. Here is research territory ripe for scientific prospectors. It is not a matter of reworking old operations, but of locating the rich new veins and perhaps in some cases merely picking up the surface nuggets.

## Bibliography


W. R. Ashby, *Design for a Brain* (New York, Wiley, 1951).

E. C. Berkeley, *Giant Brains, or Machines That Think* (New York, Wiley, 1949).

S. Butler, *Erewhon and Erewhon Revisited* (New York, Modern Library Edition, 1927).

J. Diebold, *Automation* (New York, Van Nostrand, 1952).

A. S. Householder and H. D. Landahl, *Mathematical Biophysics of the Central Nervous System* (Bloomington, Principia Press, 1945) pp. 103–110.

S. C. Kleene, *Representation of Events in Nerve Nets and Finite Automata*, Rand Corporation Memorandum RM-704, 1951.

D. M. McCallum and J. B. Smith, "Mechanized Reasoning," *Electronic Engineering* (April, 1951).

Warren S. McCulloch, and Walter Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bull. Math. Biophysics*, (1943) vol. 5, pp. 115–133.

W. S. McCulloch, "The Brain as a Computing Machine," *Electrical Engineering* (June, 1949).

John Meszar, "Switching Systems as Mechanical Brains *Bell Labs. Record*, (1953) vol. 31, pp. 63–69.

A. Oettinger, "Programming a Digital Computer to Learn," *Phil. Mag.*, (December, 1952) vol. 43, pp. 1243–1263.

W. Pease, "An Automatic Machine Tool," *Scientific American*, (September, 1952) vol. 187, pp. 101–115.

C. E. Shannon, *Presentation of a Maze-Solving Machine*, Transactions of the Eighth Cybernetics Conference, Josiah Macy, Jr. Foundation, New York, 1952, pp. 173–180.

C. E. Shannon, "Programming a Computer for Playing Chess," *Phil. Mag.*, (March, 1950) vol. 41, pp. 256–275.

C. S. Strachey, "Logical or Non-Mathematical Programmes," *Proc. of the Assn. for Computing Machinery*, Toronto (1952), pp. 46–49.

A. M. Turing, "Computing Machinery and Intelligence," *Mind*, (1950) vol. 59, pp. 433–460.

A. M. Turing, On Computable Numbers, with an Application to the Entscheidungsproblem, *Proc. Lond. Math. Soc.*, (1936) vol. 24, pp. 230–265.

J. Von Neuman, "The General and Logical Theory of Automata from Cerebral Mechanisms in Behavior," (New York, Wiley, 1951, pp. 1–41.

J. Von Neumann, *Probabilistics Logics*, California Institute of Technology, 1952.

N. Wiener, Cybernetics (New York, Wiley, 1948).


