T H E   U N I V E R S I T Y   O F   M I C H I G A N
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Department of Philosophy

First Quarterly Progress Report

1 November 1960—31 January 1961

A THEORY OF ADAPTIVE SYSTEMS

I. A LOGICAL THEORY OF ADAPTIVE SYSTEMS INFORMALLY DESCRIBED

(Precis)

J. H. Holland

ORA Project 04274

under contract with:

DEPARTMENT OF THE ARMY
U. S. ARMY SIGNAL SUPPLY AGENCY
CONTRACT NO. DA 36-039 SC-87174
FORT MONMOUTH, NEW JERSEY

administered through:

OFFICE OF RESEARCH ADMINISTRATION   ANN ARBOR

May 1961

1. INTRODUCTION

There has been increasing interest in the relationship between automata and human mental processes: also, very recently, the relationship between automata and biological growth, development, reproduction, and evolution has begun to interest researchers. Attempts are being made to formalize the study of these relationships: theories of machine adaptive systems are being developed. This paper will present one such theory of machine adaptive systems. The theory will be developed within the framework of a theory of growing automata; the particular theory will be that of "iterative circuit computers." These are computers composed of iterated modules (which contain logic and delay elements) arranged in geometric arrays.

The adaptive system theory will here be presented in four stages. (1) The _basis_ (the iterative circuit computers briefly described above); (2) the _generation_ of automata in the geometric space of iterative circuit computers; (3) _interaction_ of automata embedded in the space along with ranges of environments also embedded in the space; (4) _evaluation and accumulation_ of methods and adaptations.

2. BASIS

The theory of adaptive systems described here has as its base the class of systems called iterative circuit computers. The iterative circuit computer is _not_ itself the adaptive system; the computer is the "space" in which the adaptive system is embedded. Each iterative circuit computer is constructed of a single basic module which is iterated to form a regular array of modules. Each module has exactly the same pattern of connections to its neighbors; each module is capable of a finite number of states; each module can control the information that flows through it from its neighbors; each module can have an input to it from outside the computer. The details of the working of such computers are given in "Iterative Circuit Computers" [J. H. Holland, Proc. 1960 Western Joint Computer Conference, 259-265 (1960)]. The important characteristics for a theory of adaptive systems are: (1) the structure and arrangement of the modules of any particular class of iterative circuit computer can be completely determined by specifying certain conditions, (2) sub-programs can be written and stored at different locations throughout the computer, (3) any given growing automaton can be simulated by a connected set of sub-programs in the iterative circuit computer, (4) sub-programs can shift themselves within the computer; they can combine to form larger programs; they can produce copies of themselves in adjacent sets of modules, (5) there are methods by which any sub-program possible for a particular computer can be generated (for example, there are ways of insuring the eventual construction of any automaton), (6) relative, as opposed to absolute, addressing can be employed, (7) interpenetration, or "overwriting," of programs can be excluded if so desired.

3. RANDOM MIXING OF GENERATORS

In the theory of adaptive systems presented here, adaptation depends upon the controlled generation of methods appropriate to the environment confronting the adaptive system. The generation procedures to be described here depend upon modulation of a parallel random process whereby many programs are being generated and tested simultaneously. A simple form of random program generation, mixing, and connecting will occur when the computer is set such that (1) there is a fixed probability that the generator contents of a module will shift to one of the four neighboring modules, (2) provision is made to eliminate "conflicts" such as when a generator attempts to shift to a module already containing a generator, or two generators attempt to shift to the same module, (3) generators in adjacent modules may become connected, and afterwards shift about randomly as a unit. Since the modules themselves are completely deterministic, the "impulse" for random behavior must be set into the module from the "outside."

4. CONDITIONS FOR CONNECTION

The connection of generators is to be a function of (1) valence, and (2) activation. For any given generator the valence will be a fixed quantity throughout its history; valence specifies the "strength" and configuration of the connections the generator is permitted to make. Unlike valence, the activation associated with a generator changes with each contact made with another generator. If the sum of the activation values falls within a range fixed by their valences, then a connection is to result. By controlling the distribution of activation, the connections among generators can be controlled.

5. MODULATION OF CONNECTION RATES

The valence-activation combination provides the beginnings of a mechanism for favoring or selecting certain programs from among those which might be generated. The level of activation required for connection is completely specified by the valence of the two generators. Connections will not be directly manipulated. Instead an indirect method of control of connections is to be employed. This method of control depends upon the introduction of special strings of generators which facilitates the connection of other generator strings.

For several reasons it is important that a disconnection procedure must also be provided. If this is not done, short programs would become more and more rare, rearrangements of sub-programs of a program would be impossible, and useful "competition" between programs would be eliminated. Accordingly, rules specifying probabilistic conditions for disconnection are to be established.

6. GENERATION TREES

In the preceding sections, the means of producing and controlling the population of programs has been discussed; in this section the exercise of these controls is discussed. The programs which are generated within the automaton portion of the computer are to be used in attempts to solve problems presented by the environment portion of the underlying computer. These programs will be called trial programs to distinguish them from the programs, such as the special strings of generators, directly introduced to control the population of programs.

A "generation tree" can be employed to exhibit in detail the process whereby random movement and connection generate an unlimited variety of trial programs. The distribution of number and kind of trial programs can be skewed by introducing and producing strings of generators. The production of these strings amounts to the modulation of the parallel generation process, and has an effect far out of proportion to the number of strings produced. One reason for this is that each string takes part in connection processes repeatedly without itself being altered (a catalytic effect).

7. WELL-DEFINED PROBLEMS

Adaptation takes place relative to an automaton and its environment. Both the adapting-automaton and its environment are embedded in an iterative circuit computer. What should the environment consist of, and how is it to be presented in terms of the basic equations of the iterative circuit computer? The environment should consist of well-defined problems: a well-defined problem is presented by means of a set of initial statements and an algorithm for checking whether a purported solution is in fact a solution. These conditions are expressible within the iterative circuit computer, and if desired, can be retained there for indefinite periods not subject to alteration or internal analysis by the adapting-automaton portion of the computer. For example: let the initial statements specify an axiom system and a theorem to be proved in the system. A tentative solution will be any sequence of statements which purports to be a proof of the theorem. The checking algorithm will be the usual routine which checks that each statement in the sequence follows from previous statements by the allowed rules of inference, with the last statement being the one to be proved. It is obvious in this case that a complete knowledge of the checking procedure in no way yields a solution to the problem.

By embedding the well-defined problems in the space defined by the iterative circuit computer, in a manner analogous to the embedding of the trial programs, it is possible to carry parallel processing one step further. We can embed a whole population of problems in the space. In addition, the connected sets of generators representing the problems can be given a random motion (using the random input sequence). Thus the environment will consist of a population of well-defined problems diffusing randomly among the other connected sets of generators. (As a matter of course this random motion may be constrained in various ways by the trial programs in order to "filter out" certain problems and "process" them.) Assuming a broad enough distribution of problems, this procedure has the advantage that a more or less steady flow of results can be expected.

8. SUPERVISORY PROGRAMS AND DIFFERENTIAL SELECTION

A supervisory program can be introduced and adjusted to produce distributions of trial programs appropriate to a population of well-defined problems. The supervisory program is intended as a device to gather the threads of control at a single point; it produces a given distribution of generator strings, and as indicated earlier these strings will modulate the generation of trial programs. Thus, the supervisory program serves as an implicit definition of the distribution of trial programs at any specified time. Changes in the supervisory program will of course result in changes in the population of trial programs, and this leads in turn to the idea of differential selection of supervisory programs. The more "successful" a supervisory program is, in terms of the ability of its trial programs to produce solutions, the more predominant (in numbers) it is to become in a population of supervisory programs. If differential selection can be incorporated then the result will be an alteration of the rates of connection in the generation tree, providing a "better adaptation" of the generation process to the environment. Useless deviations within supervisory programs would, because of the disconnection rules, gradually disappear from the population. Useful deviations should by the rules, be incorporated in an ever larger proportion of supervisory programs, and would eventually be propagated throughout the computer. To implement these above notions, (1) supervisory programs must be able to duplicate themselves in order to provide successive generations to be acted upon by the selection principle, and (2) the selection principle must relate the rate of duplication of a supervisory program to the effectiveness of the trial program it produces.

These two requirements can be satisfied by a copying procedure included as a sub-program of the complete supervisory program. If the duplication is carried out by means of the introduced strings of generators, then the connections would require activation, and this in turn will "cost." It is through this "cost" that the selection principle operates. Each embedded problem will have allocated to it certain activation. If a solution (or partial solution) is obtained, it can be arranged that the activation is released, and channeled to the successful supervisory program.

9. PARTIAL SOLUTIONS AND AUTOMATIC GENERALIZATION

It may happen that a trial program will present a partial solution to a problem posed by its environment; in such an eventuality a partial release of activation can be provided for. The rules governing such partial release of activation can be incorporated in the checking routines of the problems. Such adjustments of the properties of activation released when a partial solution is obtained allows the weighting of partial solutions in relation to other partial solutions. Such adjustments can also be used to emphasize the importance of the solution of certain sets of problems over other sets.

Employing such means of adjustment and reward automata generalization becomes a characteristic of the adaptive systems here described. In all cases, the adaptive system will be "trying" continually to form a hierarchy of methods which enable it to handle larger and larger collections of problems by means of ever more general supervisory programs.

10. RICH ENVIRONMENTS

Automata generalization is accelerated when the environment is "rich": when it is composed of graded sequences of problems. The problems at a given level include problems of the next lower level as particular instances. In the natural world, such "rich" environments are not rare, and in both the natural world and in the adaptive system described here they allow the solution of problems by stages and approximations: a "leap" to a solution is not required.

11. SUBGOAL GENERATION

Subgoal generation occurs when the adaptive system divides problems into sequences of sub-problems whose solutions will contribute to the solution of the given problems. This will require that the adaptive system itself enrich the problem environment, and supply checking routines for subgoals. There are available several methods for obtaining this behavior in the system and then rewarding it for forming and satisfying the relevant subgoals. Thus, an adaptive system (composed of a hierarchy of programs) which spends a significant amount of time on problems it cannot solve will be displaced by more selective (but otherwise identical) hierarchies. Programs which can solve problems will accumulate activation at an advantageous rate, and will displace less efficient programs.

12. CONCERNING THEOREMS

It was indicated earlier (Section 2) that the structure and arrangement of the modules of any particular class of iterative circuit computers can be completely determined. By varying certain constraints we can change the class of iterative circuit computers under consideration. Statements about classes of iterative circuit computers become easier to prove or disprove as the constraints are increased and the class of computers becomes smaller. The following (familiar) strategy is suggested: apply enough constraints to allow various key statements about desirable characteristics to be proved in a relatively straightforward fashion. Then relax constraints somewhat and attempt to generalize the theorem. For example, we can consider random mixing of generators in the absence of introduced generator strings and other controls, and then proceed to cases with only one kind of template-generator-string, and so on.

Some problems on which theorems might be possible are: (1) what kinds of constraints are sufficient to secure equilibrium or stable composition of programs? What kinds of constraints are sufficient to assure this <u>cannot</u> happen? (2) When new factors are added to stable population, under what conditions will the population again reach equilibrium?

Many other problem areas (and thus areas for theorem-proving) exist.

13. GENERAL COMMENT

The theory outlined here makes considerable use of probabilistic and random processes. It seems likely that these probabilistic processes are not absolutely necessary, and that deterministic processes would suffice. Random procedures do, however, greatly simplify the formulation.

DISTRIBUTION LIST

(one copy unless otherwise noted)

OASD (R and D), Room 3E1065
The Pentagon
Washington 25, D. C.
Attn: Technical Library

Chief of Research and Development
OCS, Dept. of the Army
Washington 25, D. C.

Chief Signal Officer (2)
Department of the Army
Washington 25, D. C.
Attn: SIGPD-8b1

Chief Signal Officer
Department of the Army
Washington 25, D. C.
Attn: SIGRD

Director
U. S. Naval Research Laboratory
Washington 25, D. C.
Attn: Code 2027

Commanding Officer and Director
U. S. Navy Electronics Laboratory
San Diego 52, California

Commander
Wright Air Development Division
Wright-Patterson Air Force Base
Ohio
Attn: WCOSI-3

Commander
Air Force Cambridge Research Center
L. G. Hanscomb Field
Bedford, Massachusetts
Attn: CROTR

Commander
Rome Air Development Center
Air Research and Development Command
Griffiss Air Force Base,
New York
Attn: RCSSLD

Commanding Officer
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey
Attn: Logistics Division

Commander (7)
Armed Services Technical
Information Agency
Arlington Hall Station
Arlington 12, Virginia
Attn: TIPDR

Commanding Officer
U. S. Army Signal Equipment
Support Agency
Fort Monmouth, New Jersey
Attn: SIGFM/ES-ADJ

U. S. Continental Army
Command Liaison Office
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey

Corps of Engineers Liaison Office
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey

ARDC Liaison Office
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey

DISTRIBUTION LIST (Continued)

U. S. Navy Electronics
Liaison Office
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey

Marine Corps Liaison Office
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey

Commanding Officer
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey
Attn: Director of Research/Engineering
Attn: Technical Documents Center
Attn: SIGRA/SL-ADJ
Attn: Technical Information Division (4)

Commanding Officer
U. S. Army Electronic Proving Ground
Fort Huachuca, Arizona
Attn: Automatic Data Processing
Department

Chief Signal Officer
Department of the Army
Washington 25, D. C.
Attn: SIGRD-6B

Commanding Officer (2)
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey
Attn: Exploratory Research Division
"C", Jacob Borsuk

Signal Corp Liaison Officer
Rome Air Development Center (RAOL)
Griffiss Air Force Base,
New York

U. S. Air Force Security Service
San Antonio, Texas
Attn: Major W. W. Fries

Commander, Rome Air Development
Center
Air Research and Development Command
Griffiss Air Force Base,
New York
Attn: RCWID, I. Iuorno

Commanding Officer
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey
Attn: Exploratory Research,
Dr. Reilly
Attn: Exploratory Research Math
Division, Dr. Babbitt
Attn: Engineering Sciences Dept.,
Mr. Hennessy

Director
National Bureau of Standards
U. S. Department of Commerce
Washington 25, D. C.
Attn: Dr. Alexander

Office of Naval Research
Department of the Navy
Washington 25, D. C.
Attn: Dr. Tavitz

Commander
Wright Air Development Division
Wright-Patterson Air Force Base
Dayton, Ohio
Attn: Dr. Steele

Chief, Bureau of Ships
Washington 25, D. C.
Attn: Donald Rheem

Director
National Security Agency
Fort George C. Meade,
Maryland
Attn: RADE 32

DISTRIBUTION LIST (Concluded)

Commanding Officer
Army Pictorial Center
35-11 35th Street
Long Island City, New York
Attn: Mr. Erwin Oeller

Commanding Officer (6)
U. S. Army Signal Research and
Development Laboratory
Fort Monmouth, New Jersey
Attn: SIGRA/SL-NPE

Professor Heinz Von Foerster
215 Electrical Engineering Research Laboratory
University of Illinois
Urbana, Illinois

Director
National Security Agency
Fort George G. Meade, Maryland
Attn: CREF-141 (Room 2C087),
Miss Creswell