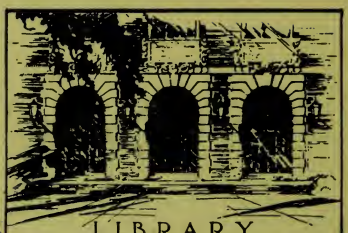
THEORY OF SELF-REPRODUCING AUTOMATA



Commerce

Theory of
Self-Reproducing Automata

Theory of
Self-Reproducing Automata

JOHN VON NEUMANN

edited and completed by Arthur W. Burks

University of Illinois Press
URBANA AND LONDON 1966

Manufactured in the United States of America. Library of Congress Catalog Card No. 63-7246.

CONTENTS

CHAPTER 2

A SYSTEM OF 29 STATES WITH A GENERAL TRANSITION RULE

CHAPTER 3

DESIGN OF SOME BASIC ORGANS

CHAPTER 4

DESIGN OF A TAPE AND ITS CONTROL

CHAPTER 5

[AUTOMATA SELF-REPRODUCTION]

TABLES

LIST OF FIGURES

Second Lecture

Fifth Lecture

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

PREFACE

In the late 1940's John von Neumann began to develop a theory of automata. He envisaged a systematic theory which would be mathematical and logical in form and which would contribute in an essential way to our understanding of natural systems (natural automata) as well as to our understanding of both analog and digital computers (artificial automata).

To this end von Neumann produced five works, in the following order:

(1) "The General and Logical Theory of Automata." Read at the Hixon Symposium in September, 1948; published in 1951. *Collected Works* 5.288–328.[1]

(2) "Theory and Organization of Complicated Automata." Five lectures delivered at the University of Illinois in December, 1949. This is Part I of the present volume.

(3) "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." Lectures given at the California Institute of Technology in January, 1952. *Collected Works* 5.329–378.

(4) "The Theory of Automata: Construction, Reproduction, Homogeneity." Von Neumann started this manuscript in the fall of 1952 and continued working on it for about a year. This is Part II of the present volume.

(5) *The Computer and the Brain.* Written during 1955 and 1956; published in 1958.

The second and fourth of these were left at his death in a manuscript form which required extensive editing. As edited they constitute the two parts of the present volume, which thus concludes von Neumann's work on the theory of automata.

As a background for this editorial work I made a study of all of von Neumann's contributions on computers, including the theory of automata. I have summarized his contributions in the "Introduction" to the present volume.

Von Neumann was especially interested in complicated automata, such as the human nervous system and the tremendously large com-

[1] Complete references are given in the bibliography. "*Collected Works* 5.288–328" refers to pp. 288–328 of Vol. V of von Neumann's *Collected Works*.

puters he foresaw for the future. He wanted a theory of the logical organization of complicated systems of computing elements and believed that such a theory was an essential prerequisite to constructing very large computers. The two problems in automata theory that von Neumann concentrated on are both intimately related to complexity. These are the problems of reliability and self-reproduction. The reliability of components limits the complexity of the automata we can build, and self-reproduction requires an automaton of considerable complexity.

Von Neumann discussed reliability at length in his "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." His work on self-reproducing automata is found chiefly in the present volume. Part II, which constitutes the bulk of the present volume, treats the logical design of a self-reproducing cellular automaton. Though the shorter Part I is devoted to complicated automata in general, its high point is the kinematic model of self-reproduction (Fifth Lecture). It therefore seemed appropriate to use the title "Theory of Self-Reproducing Automata" for the whole work.

It is unfortunate that, because of his premature death, von Neumann was unable to put in final form any of the research he was doing in automata theory. The manuscripts for both parts of the present volume were unfinished; indeed, they were both, in a sense, first drafts. There is one compensation in this: one can see von Neumann's powerful mind at work. Early drafts by a thinker of von Neumann's ability are not often available. For this reason, I have tried hard to preserve the original flavor of von Neumann's manuscripts, while yet rendering them easily readable. So that the reader will know what the raw manuscripts are like, I will describe them and the editorial changes I have made in them.

Von Neumann agreed to write a book on automata theory in connection with his five lectures at the University of Illinois in December, 1949. A tape recording of the lectures was made to aid him in writing the book. Unfortunately, the recording and the typescript of it turned out badly, with gaps in the text, unintelligible passages, and missing words. Von Neumann himself never edited this typescript, but instead planned to use the manuscript "The Theory of Automata: Construction, Reproduction, Homogeneity" for the promised book. The recording itself is not extant. Despite these circumstances, the Illinois lectures deserve publication, and the recorded version, highly edited of necessity, constitutes Part I of this volume.

Von Neumann prepared a detailed outline of the lectures in advance

of delivery, and the content of the lectures corresponded roughly to this outline. The outline bore the title "Theory and Organization of Complicated Automata," and began with the following three lines:

> The logical organization and limitations of high-speed digital computers.
>
> Comparison of these and other complicated automata, both artificial and natural.
>
> Inference from the comparison of the nervous systems found in nature.

There followed the title of each lecture and a list of topics to be covered in that lecture; these are reproduced verbatim at the beginning of each lecture below, even though the lecture materials do not correspond exactly to the list of topics.

Because of the state of the manuscript it has been necessary to do much editing. The typescript of the recording is especially poor in the more formal portions of the lectures, where von Neumann used the blackboard. For these portions, particularly, I have found two sets of notes taken at the lectures to be helpful. I have preserved von Neumann's phraseology where feasible, but I have frequently found it necessary to use my own words. I have sometimes felt it best to summarize what von Neumann was saying rather than to attempt to reconstruct the text. Several of the points von Neumann made in the Illinois lectures also appear in his published writings, or are well known, and in these cases I have often summarized what von Neumann said or given references to his published works.

Where the writing is strictly my own, it appears in brackets. The reconstructed edition of von Neumann's words is not bracketed, but it should be kept in mind that much of this unbracketed text is heavily edited.

The manuscript "The Theory of Automata: Construction, Reproduction, Homogeneity" was in a much better state. It seems to have been a first draft, with the exception that there is an earlier outline (with figures) of the procedure whereby the memory control **MC** lengthens and shortens the connecting loop $\mathbf{C}_1$ and the timing loop $\mathbf{C}_2$ under the direction of the constructing unit **CU** (cf. Secs. 4.1 and 4.2). Despite its being a first draft, the manuscript was publishable as it stood except for deficiencies of the following three types.

(1) First, the manuscript lacked many of those simple mechanisms which make for easy reading. There were no figure titles. Formulas, sections, and figures were referred to by number only, without explicit indication as to whether the item referred to is a formula, section, or figure. Section titles were listed on a separate sheet. Also

on a separate sheet, von Neumann gave only brief indications for the footnotes he planned. Organs were referred to by letter alone. For example, von Neumann merely used "A" and "B" to refer to the organs I have called the "constructing unit **CU**" and the "memory control **MC**," respectively. I have worked through the manuscript several times and each time I have been amazed at how von Neumann could keep track of what he was doing with so few mnemonic devices.

In editing the manuscript I have endeavored to supply these devices. For example, where von Neumann wrote "**CO**" I often put "control organ **CO**." I have added titles to the figures and completed the footnote references. Von Neumann wrote some explanatory remarks on the figures; these have been moved to the text. Similar and related changes have been made, all without any indication in the text.

In addition, I have inserted footnotes, commentaries, explanations, and summaries at various places in the text, and have added a concluding chapter (Ch. 5). All such additions are in brackets. Von Neumann's brackets have been changed to braces, except for his usage "[0]" and "[1]" to refer to ordinary and special symbols. In connection with my bracketed additions, I have added Tables I and V and many figures. Figures 1–8, 16, 18, 19, 22, 24, 28–36, 38, 39, and 41 are von Neumann's; the remaining figures are mine.

(2) Second, the manuscript "Theory of Automata: Construction, Reproduction, Homogeneity" contained many errors. These range from minor slips (which I have corrected without any specific indication), through errors of medium significance (which I have corrected or commented on in bracketed passages), to major errors requiring considerable redesign (which I have discussed in Sections 5.1.1 and 5.1.2). All of these errors are correctable, but because organs designed in the earlier parts of the manuscript are used in later parts, many of these errors propagate and "amplify." In this connection, it should be kept in mind that the manuscript was an early draft, and that von Neumann was working out the design as he proceeded, leaving many design parameters for later specification.

(3) Third, the manuscript "Theory of Automata: Construction, Reproduction, Homogeneity" is incomplete. The construction stops before the tape unit is quite finished. In Chapter 5 I show how to complete the design of von Neumann's self-reproducing automaton.

The technical development of the manuscript is extremely complicated and involved. The deficiencies just mentioned add to its difficulty. In some respects it would have been editorially easier not to edit the manuscript after Chapter 2 and instead work out the

design of von Neumann's self-reproducing automaton along the lines he last envisaged. But this was not a real alternative because of the historical importance of the manuscript, and the opportunity it gives to observe a powerful mind at work. I have therefore endeavored to make corrections and add comments so as to preserve the original style of the manuscript while making it relatively easy to read.

I am indebted to a number of people for their assistance. The late Mrs. Klara von Neumann-Eckardt gave me information about her husband's manuscripts. Several people who worked with von Neumann on computers gave me firsthand information: Abraham Taub, Herman Goldstine, the late Adele Goldstine, and especially Julian Bigelow and Stan Ulam; von Neumann often discussed his work on automata theory with Bigelow and with Ulam. John Kemeny, Pierce Ketchum, E. F. Moore, and Claude Shannon heard lectures by or had discussions with von Neumann on automata theory. Kurt Gödel's letter at the end of the Second Lecture of Part I is reproduced with his kind permission. Thanks go to many of my graduate students and research associates for technical assistance, particularly Michael Faiman, John Hanne, James Thatcher, Stephen Hedetniemi, Frederick Suppe, and Richard Laing. Alice Finney, Karen Brandt, Ann Jacobs, and Alice R. Burks have provided editorial assistance. M. Elizabeth Brandt drew the figures. My editorial work was supported by the National Science Foundation. None of these share any responsibility for the editing.

Arthur W. Burks
Ann Arbor, 1965

EDITOR'S INTRODUCTION

Von Neumann's Work on Computers

John von Neumann was born on December 28, 1903 in Budapest, Hungary, and died in Washington, D.C., February 8, 1957.[1] He earned a doctorate in mathematics from the University of Budapest and an undergraduate chemistry degree from the Eidgenössische Technische Hochschule in Zurich, Switzerland. He became a Privatdocent at the University of Berlin in 1927 and a Privatdocent at the University of Hamburg in 1929. In 1930 he came to the United States as a visiting lecturer at Princeton University, where he was made full professor in 1931. In 1933 he joined the newly formed Institute for Advanced Study as a professor and retained that post for the rest of his life.[2]

In later life, while retaining his theoretical interests and productivity, von Neumann developed strong interests in the applications of mathematics. During the Second World War he became heavily involved in scientific research on problems of defense. He played a major role in the development of the atomic bomb, contributing particularly to the method of implosion. He was a consultant to many government laboratories and organizations and a member of many important scientific advisory committees. After the war he continued these consulting and advisory activities. Altogether he was involved in such diverse fields as ordnance, submarine warfare, bombing objectives, nuclear weapons (including the hydrogen bomb), military strategy, weather prediction, intercontinental ballistic missiles, high-speed electronic digital computers, and computing methods. In October, 1954, the President of the United States appointed him to the United States Atomic Energy Commission, a position he held at the time of his death. He received many awards and honors during his lifetime, including membership in the National Academy of Sciences, two Presidential Awards, and the Enrico Fermi Award of the Atomic Energy Commission. The latter was given especially for his

[1] See Ulam, "John von Neumann," and Mrs. von Neumann's preface to *The Computer and the Brain.*

[2] See his *Collected Works*, edited by A. Taub. An excellent summary of von Neumann's accomplishments is presented in the *Bulletin of the American Mathematical Society*, Vol. 64, No. 3, Part 2, May, 1958.

contributions to the development of electronic computers and their uses.

Von Neumann the Mathematician. During the last years of his life John von Neumann devoted considerable effort to developing a theory of automata. The present volume, edited from two unfinished manuscripts, is his last work on this subject. Because of his premature death he was unable to finish a volume which would present a complete picture of what he wished to accomplish. It is therefore appropriate to summarize here the main features of his projected theory of automata. Since his conception of automata theory arose out of his work in mathematics and computers, we will begin by describing that work.

Von Neumann was a very great mathematician. He made many important contributions in a wide range of fields. Von Neumann himself thought his most important mathematical achievements were in three areas: the mathematical foundations of quantum theory, the theory of operators, and ergodic theory. His contributions in other areas bear more directly on his computer work. In the late 1920's he wrote on symbolic logic, set theory, axiomatics, and proof theory. In the middle thirties he worked on lattice theory, continuous geometry, and Boolean algebra. In a famous paper of 1928 and in a book of 1944[3] he founded the modern mathematical theory of games. Starting in the late thirties and continuing through and after the war he did much research in fluid dynamics, dynamics, problems in the mechanics of continua arising out of nuclear technology, and meteorology. During the war he became involved in computing and computers, and after the war this became his main interest.

Von Neumann and Computing. Von Neumann was led into computing by his studies in fluid dynamics. Hydrodynamical phenomena are treated mathematically by means of non-linear partial differential equations. Von Neumann became especially interested in hydrodynamical turbulence and the interaction of shock waves. He soon found that existing analytical methods were inadequate for obtaining even qualitative information about the solutions of non-linear partial differential equations in fluid dynamics. Moreover, this was so of non-linear partial differential equations generally.

Von Neumann's response to this situation was to do computing.[4] During the war he found computing necessary to obtain answers to

[3] "Zur Theorie der Gesellschaftsspiel." *Theory of Games and Economic Behavior*, with Oskar Morgenstern.

[4] See Ulam, "John von Neumann," pp. 7–8, 28 ff., and Birkhoff, *Hydrodynamics*, pp. 5, 25.

problems in other fields, including nuclear technology. Hence, when the new high-speed electronic digital general-purpose computers were developed during and after the war, he was quick to recognize their potentialities for hydrodynamics as well as other fields. In this connection he developed a general method for using computers which is of very great importance because it is applicable to a wide variety of problems in pure and applied mathematics.

The procedure which he pioneered and promoted is to employ computers to solve crucial cases numerically and to use the results as a heuristic guide to theorizing. Von Neumann believed experimentation and computing to have shown that there are physical and mathematical regularities in the phenomena of fluid dynamics and important statistical properties of families of solutions of the non-linear partial differential equations involved. These regularities and general properties could constitute the basis of a new theory of fluid dynamics and of the corresponding non-linear equations. Von Neumann believed that one could discover these regularities and general properties by solving many specific equations and generalizing the results. From the special cases one would gain a feeling for such phenomena as turbulence and shock waves, and with this qualitative orientation could pick out further critical cases to solve numerically, eventually developing a satisfactory theory. See the First Lecture of Part I of this volume.

This particular method of using computers is so important and has so much in common with other, seemingly quite different, uses of computers that it deserves extended discussion. It is of the essence of this procedure that computer solutions are not sought for their own sake, but as an aid to discovering useful concepts, broad principles, and general theories. It is thus appropriate to refer to this as the *heuristic use* of computers.[5]

The heuristic use of computers is similar to and may be combined with the traditional hypothetical-deductive-experimental method of science. In that method one makes an hypothesis on the basis of the available information, derives consequences from it by means of mathematics, tests the consequences experimentally, and forms a new hypothesis on the basis of the findings; this sequence is iterated indefinitely. In using a computer heuristically one proceeds in the same way, with computation replacing or augmenting experimentation. One makes an hypothesis about the equations under investiga-

[5] See also Ulam, *A Collection of Mathematical Problems*, Ch. 8, "Computing Machines as a Heuristic Aid."

tion, attempts to pick out some crucial special cases, uses a computer to solve these cases, checks the hypothesis against the results, forms a new hypothesis, and iterates the cycle.

The computations may also be compared with experimental data. When this is done the heuristic use of computers becomes simulation. Computation in itself can only provide answers to purely mathematical questions, so when no comparison is made with empirical fact the heuristic use of computers contributes to pure mathematics. Von Neumann thought that the main difficulties in fluid dynamics stemmed from inadequate mathematical knowledge of non-linear partial differential equations, and that the heuristic use of computers would help mathematicians to construct an adequate and useful theory for this subject. He pointed out that while much progress had been made by means of wind tunnels, since the equations governing the phenomena were known, these wind tunnels were being used as analog computers rather than as experimental apparatus.

> . . . many branches of both pure and applied mathematics are in great need of computing instruments to break the present stalemate created by the failure of the purely analytical approach to non-linear problems. . . . really efficient high-speed computing devices may, in the field of non-linear partial differential equations as well as in many other fields, which are now difficult or entirely denied access, provide us with those heuristic hints which are needed in all parts of mathematics for genuine progress.[6]

Von Neumann's suggestion that powerful computers may provide the mathematician "with those heuristic hints which are needed in all parts of mathematics for genuine progress" is connected to his strong conviction that pure mathematics depends heavily on empirical science for its ideas and problems. ". . . the best inspirations of modern mathematics . . . originated in the natural sciences." [7] He recognized that mathematics is not an empirical science and held that the mathematician's criteria of selection of problems and of success are mainly aesthetical.

> I think that it is a relatively good approximation to truth—which is much too complicated to allow anything but approximations—that mathematical ideas originate in empirics, although the genealogy is sometimes long and obscure. But once they are so conceived, the subject begins to live a peculiar life of its own and is better compared to a creative one, governed by almost entirely aesthetical motivations, than to anything else and, in par-

[6] Von Neumann and Goldstine, "On the Principles of Large Scale Computing Machines," *Collected Works* 5.4.

[7] "The Mathematician," *Collected Works* 1.2. The next quotation is from the same article, 1.9.

ticular, to an empirical science. There is, however, a further point which, I believe, needs stressing. . . . at a great distance from its empirical source, or after much "abstract" inbreeding, a mathematical subject is in danger of degeneration. . . . whenever this stage is reached, the only remedy seems to me to be the rejuvenating return to the source: the reinjection of more or less directly empirical ideas.

The role that empirical science plays in pure mathematics is a heuristic one: empirical science supplies problems to investigate and suggests concepts and principles for their solution. While von Neumann never said so, I think it likely that he thought the computations produced by the heuristic use of computers can play the same role in some areas of mathematics. In the First Lecture of Part I below he said that powerful methods in pure mathematics depend for their success on the mathematicians having an intuitive and heuristic understanding of them, and suggested that one can build this intuitive familiarity with non-linear differential equations by using computers heuristically.[8]

It should be noted that in the heuristic use of computers the human, not the machine, is the main source of suggestions, hypotheses, heuristic hints, and new ideas. Von Neumann wished to make the machine as intelligent as possible, but he recognized that human powers of intuition, spatial imagery, originality, etc., are far superior to those of present or immediately foreseeable machines. He wished to augment the ability of a skilled, informed, creative human by the use of a digital computer as a tool. This procedure would involve considerable interaction between man and the machine and would be facilitated by automatic programming and by input-output equipment designed for direct human use.

Once he became interested in computing, von Neumann made important contributions to all aspects of the subject and its technology. The extant methods of computation had been developed for hand computation and punched card machines and hence were not well suited to the new electronic computers, which were several orders of magnitude faster than the old. New methods were needed, and von Neumann developed many of them. He contributed at all levels. He devised algorithms and wrote programs for computations ranging from the calculation of elementary functions to the integration of non-linear partial differential equations and the solutions of games.

[8] In view of von Neumann's emphasis on the role of intuition in mathematical discovery it is of interest to note that von Neumann's own intuition was auditory and abstract rather than visual. See Ulam, "John von Neumann, 1903–1957," pp. 12, 23, and 38–39.

He worked on general techniques for numerical integration and inverting matrices. He obtained results in the theory of numerical stability and the accumulation of round-off errors. He helped develop the Monte Carlo method for solving integro-differential equations, inverting matrices, and solving linear systems of equations by random sampling techniques.[9] In this method the problem to be solved is reduced to a statistical problem which is then solved by computing the results for a sufficiently large sample of instances.

Von Neumann also made important contributions to the design and programming of computers, and to the theory thereof. We will survey his work in these areas next.

Logical Design of Computers. With his strong interest in computing and his background in logic and physics it was natural for von Neumann to become involved in the development of high-speed electronic digital computers. The first such computer was the ENIAC, designed and built at the Moore School of Electrical Engineering of the University of Pennsylvania during the period 1943 to 1946.[10] Von Neumann had some contacts with this machine, and so a few words about it are in order.

The idea of constructing a general purpose high-speed computer of electronic components originated with John Mauchly, who suggested to H. H. Goldstine of the Ordnance Department that the United States Army support the development and construction of such a machine, to be used primarily for ballistics computations. This support was given, the Army being impressed especially with the great speed with which an electronic computer could prepare firing tables. The ENIAC was designed and constructed by a number of people, including the writer, under the technical direction of Mauchly and J. P. Eckert. Von Neumann came to visit us while we were building the ENIAC, and he immediately became interested in it. By this time the design of the ENIAC was already fixed, but after the ENIAC was completed von Neumann showed how to modify it so that it was much simpler to program. In the meantime he developed the logical design for a radically new computer, which we will describe later.

The ENIAC was, of course, radically different from any earlier

[9] Ulam, "John von Neumann," pp. 33–34. Von Neumann *Collected Works* 5.751–764. The method is described in Metropolis and Ulam, "The Monte Carlo Method."

[10] See Burks, "Electronic Computing Circuits of the ENIAC" and "Super Electronic Computing Machine," Goldstine and Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," and Brainerd and Sharpless, "The ENIAC."

computer, but interestingly enough, it was also quite different from its immediate successors. It differed from its immediate successors in two fundamental respects: the use of several semiautonomous computing units working simultaneously and semi-independently, and the exclusive reliance on vacuum tubes for high-speed storage. Both of these design features resulted from the electronic technology of the time.

The basic pulse rate of the ENIAC circuits was 100,000 pulses per second. To obtain a high computation speed all 10 (or 20) decimal digits were processed in parallel, and, moreover, a large number of computing units were constructed, each with some local programming equipment, so that many computations could proceed simultaneously under the overall direction of a master programming unit. There were 30 basic units in the ENIAC: 20 accumulators (each of which could store and add a 10-digit number), 1 multiplier, 1 divider and square-rooter, 3 function table units, an input unit, an output unit, a master programmer, and 2 other units concerned with control. All of these basic units could operate at the same time.

At that time the vacuum tube was the only reliable high-speed storage device—acoustic delay lines, electrostatic storage systems, magnetic cores, etc., all came later—and so of necessity vacuum tubes were used for high-speed storage as well as for arithmetic and for logical control. This entailed a severe limitation on the high-speed store, as vacuum tubes are an expensive and bulky storage medium—the ENIAC contained 18,000 vacuum tubes as it was, a sufficient number for the skeptics to predict that it would never operate properly. The limited high speed storage of 20 10-digit decimal numbers was augmented by large quantities of low-speed storage of various types: electromagnetic relays for input and output, hand operated mechanical switches controlling resistor matrices in the function table units for the storage of arbitrary numerical functions and of program information, and hand-operated mechanical switches and flexible plug-in cables for programming.

A general purpose computer must be programmed for each particular problem. This was done on the ENIAC by hand: by setting mechanical switches of the program controls of each of the computing units used in the problem, interconnecting these program controls with cables, and setting the switches of the function tables. This programming procedure was long, laborious, and hard to check, and while it was being done the machine stood idle. After the ENIAC was completed, von Neumann showed how to convert it into a centrally programmed computer in which all the programming could be done

by setting switches on the function tables. Each of the three function table units had a switch storage capacity of 104 entries, each entry consisting of 12 decimal digits and 2 sign digits. However, the pulses used to represent numbers were the same size and shape as the pulses used to stimulate program controls, so that the function table units could also be used to store program information. On von Neumann's scheme the outputs of the function tables were connected to the program controls of the other units through some special equipment and the master programmer, and the switches on the program controls of these units were set. All of this was done in such a way that it need not be changed from problem to problem. Programming was thus reduced to setting switches by hand on the function table units.

In the meantime we were all concerned with the design of much more powerful computers. As mentioned earlier, the greatest weakness of the ENIAC was the smallness of its high-speed storage capacity, resulting from the technological fact that at the time the design of the ENIAC was fixed the vacuum tube was the only known reliable high-speed storage component. This limitation was overcome and the technology of computers changed abruptly when J. P. Eckert conceived of using an acoustic delay line as a high-speed storage device. Acoustic delay lines made of mercury had been used to delay pulses in war time radar equipment. Eckert's idea was to feed the output of a mercury delay line (through an amplifier and pulse reshaper) back into its input, thereby storing a large number of pulses in a circulating memory. A circulating memory of, say, 1000 bits could be built with a mercury delay line and a few tubes, in contrast to the ENIAC where a double triode was required for each bit.

In the ENIAC the few numbers being processed were stored in circuits that could be changed both automatically and rapidly; all other numbers and the program information were stored in electromagnetic relays, switches, and cable interconnections. It now became possible to store all this information in mercury delay lines, where it would be quickly and automatically accessible. The ENIAC was a mixed synchronous, asynchronous machine. The use of pulses in the mercury delay lines made it natural to build a completely synchronous machine timed by a central source of pulses called the "clock." Eckert and Mauchly designed circuits capable of operating at a pulse rate of 1 megacycle, 10 times the basic pulse rate of the ENIAC, and gave considerable thought to the design of a mercury delay line machine. Goldstine brought von Neumann in as a consultant, and we all participated in discussions of the logical design of such a machine. It was decided to use the binary system. Since the delay lines operated

serially, the simplest way to process the bits was seriatim. All of this made it possible to build a machine much smaller than the ENIAC and yet much more powerful than the ENIAC. The proposed machine was to be called the EDVAC. It was estimated that it could be built with about 3000 vacuum tubes.

Von Neumann then worked out in considerable detail the logical design of this computer. The result appeared in his *First Draft of a Report on the EDVAC*,[11] which was never published. Since this report contained the first logical design of an electronic computer in which the program could be stored and modified electronically, I will summarize its contents. Of particular interest to us here are the following features of his design: the separation of logical from circuit design, the comparison of the machine to the human nervous system, the general organization of the machine, and the treatment of programming and control.

Von Neumann based his construction on idealized switch-delay elements derived from the idealized neural elements of McCulloch and Pitts.[12] Each such element has one to three excitatory inputs, possibly one or two inhibitory inputs, a threshold number (1, 2, 3), and a unit delay. It emits a stimulus at time $t + 1$ if and only if two conditions are satisfied at time t: (1) no inhibitory input is stimulated, (2) the number of excitatory inputs stimulated is at least as great as the threshold number.[13]

The use of idealized computing elements has two advantages. First, it enables the designer to separate the logical design from the circuit design of the computer. When designing the ENIAC, we developed logical design rules, but these were inextricably tied in with rules governing circuit design. With idealized computing elements one can distinguish the purely logical (memory and truth-functional) requirements for a computer from the requirements imposed by the state of technology and ultimately by the physical limitations of the materials and components from which the computer is made. Logical design is the first step; circuit design follows. The elements for logical design

[11] The initials abbreviate "Electronic Discrete Variable Automatic Computer." The machine of this name actually constructed at the Moore School of Electrical Engineering was built after the people mentioned above were no longer connected with the Moore School. The logical design of the Cambridge University EDSAC was based on this report. Wilkes, "Progress in High-Speed Calculating Machine Design" and *Automatic Digital Computers*.

[12] "A Logical Calculus of the Ideas Immanent in Nervous Activity."

[13] The threshold elements of "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Collected Works* 5.332, are similar, but differ with respect to the operation of inhibitory inputs.

should be chosen to correspond roughly with the resultant circuits; that is, the idealization should not be so extreme as to be unrealistic.

Second, the use of idealized computing elements is a step in the direction of a theory of automata. Logical design in terms of these elements can be done with the rigor of mathematical logic, whereas engineering design is necessarily an art and a technique in part. Moreover, this approach facilitates a comparison and contrast between different types of automata elements, in this case, between computer elements on the one hand and neurons on the other. Von Neumann made such comparisons in *First Draft of a Report on the EDVAC*, noting the differences as well as the similarities. Thus he observed that the circuits of the EDVAC were to be synchronous (timed by a pulse clock) while the nervous system is presumably asynchronous (timed autonomously by the successive reaction times of its own elements). He also noted the analogy between the associative, sensory, and motor neurons of the human nervous system on the one hand, and the central part of the computer, its input, and its output, respectively. This comparison of natural and artificial automata was to become a strong theme of his theory of automata.

The organization of the EDVAC was to be radically different from that of the ENIAC. The ENIAC had a number of basic units, all capable of operating simultaneously, so that many streams of computation could proceed at the same time. In contrast, the proposed EDVAC had only one basic unit of each kind, and it never performed two arithmetical or logical operations simultaneously. These basic units were a high-speed memory **M**, a central arithmetic unit **CA**, an outside recording medium **R**, an input organ **I**, an output organ **O**, and a central control **CC**.

The memory **M** was to be composed of possibly as many as 256 delay lines each capable of storing 32 words of 32 bits each, together with the switching equipment for connecting a position of **M** to the rest of the machine. The memory was to store initial conditions and boundary conditions for partial differential equations, arbitrary numerical functions, partial results obtained during a computation, etc., as well as the program (sequence of orders) directing the computation. The outside recording medium **R** could be composed of punched cards, paper tape, magnetic wire or tape, or photographic film, or combinations thereof. It was to be used for input and output, as well as for auxiliary low-speed storage. The input organ **I** transferred information from **R** to **M**; the output organ **O** transferred information from **M** to **R**. The notation of **M** was binary; that of **R** was decimal.

The central arithmetic unit **CA** was to contain some auxiliary regis-

ters (one-word delay lines) for holding numbers. Under the direction of the central control **CC** it was to add, subtract, multiply, divide, compute square-roots, perform binary-decimal and decimal-binary conversions, transfer numbers among its registers and between its registers and **M**, and choose one of two numbers according to the sign of a third number. The last operation was to be used for transfer of control (jumping conditionally) from one order in the program to another. Numbers were processed in **CA** serially, the least significant bits being treated first, and only one operation was performed at a time.

The first bit of each word was zero for a number, one for an order. Eight bits of an order were allotted to the specification of the operation to be performed and, if a reference to **M** was required, thirteen bits to an address. A typical sequence would go like this. Suppose an addition order with memory address x was located in position y of **M**, the addend in the next position $y + 1$, and the next order to be executed in the next position $y + 2$. The order at y would go into **CC**, the addend at $y + 1$ into **CA**, and the augend would be found in **CA**; the sum would be placed in position x of **M**. The order at position $y + 2$ would be executed next.

Normally orders were taken from the delay lines in sequence, but one order with address z provided for **CC** to take its next order from memory position z. When a number was transferred from **CA** to address w of **M**, account was taken of the contents of w; if w contained an order (i.e., a word whose first bit was one), then the 13 most significant bits of the result in **CA** were substituted for the 13 address bits located in w. The addresses of orders could be modified automatically by the machine in this way. This provision, together with the order for shift of control to an arbitrary memory position w and the power of **CA** to choose one of two numbers according to the sign of a third, made the machine a fully automatic stored program computer.

At the same time that he worked out the logical design of the EDVAC von Neumann suggested the development of a high-speed memory incorporating the principle of the iconoscope.[14] Information is placed on the iconoscope by means of light and sensed by an electron beam. Von Neumann suggested that information could also be placed on the inside surface of such a tube by means of an electron beam. The net result would be storage in the form of electrostatic charges on a dielectric plate inside a cathode-ray tube. He predicted that such a

[14] *First Draft of a Report on the EDVAC*, Section 12.8.

memory would prove superior to the delay line memory. It soon became apparent that this was so, and von Neumann turned his attention to an even more powerful computer based on such a memory.

The new computer was to be much faster than any other machine under consideration, mainly for two reasons. First, in an electrostatic storage system each position is immediately accessible, whereas a bit or word stored in a delay line is not accessible until it travels to the end of the line. Second, it was decided to process all (40) bits of a word in parallel, thereby reducing the computation time. The logical design is given in *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument.*[15] The proposed computer was built at the Institute for Advanced Study by a number of engineers under the direction of Julian Bigelow, and was popularly known as the JONIAC.[16] While the machine was still under construction, its logical and circuit design was influential on many computers constructed in the United States, including computers at the University of Illinois, Los Alamos National Laboratory, Argonne National Laboratory, Oak Ridge National Laboratory, and the Rand Corporation, as well as some machines produced commercially. The JONIAC played an important role in the development of the hydrogen bomb.[17]

Programming and Flow Diagrams. Von Neumann immediately recognized that these new computers could solve large problems so fast that new programming procedures would be needed to enable mathematicians and programmers to make full use of the powers of these machines. With the order code of the proposed Institute for Advanced Study computer in mind he proceeded to develop new programming methods. The results were presented in the influential series of reports *Planning and Coding of Problems for an Electronic Computing Instrument.*[18]

One generally begins with a mathematical formulation of a problem and then decides what explicit computational methods he will employ. These methods are almost always highly inductive, involving recursions within recursions many times over. What one has at this

[15] This was written in collaboration with H. H. Goldstine and the present writer. It was typical of von Neumann that he wanted the patentable material in this report to belong to the public domain, and at his suggestion we all signed a notarized statement to this effect.

[16] It is described by Estrin, "The Electronic Computer at the Institute for Advanced Study." The original plan was to use the memory described by Rajchman in "The Selectron—a Tube for Selective Electrostatic Storage," but the actual memory consisted of cathode-ray tubes operated in the manner described by Williams in "A Cathode-Ray Digit Store."

[17] *New York Times*, Feb. 9, 1957, p. 19.

[18] Written in collaboration with H. H. Goldstine.

stage is a general description of the desired computation, expressed in the ordinary language and mathematical symbolism of the mathematician. The task is now to transform this description into a program expressed in machine language. This is not a simple, straightforward translation task, however, partly because of the generality of the description of the computation and partly because of the nature of recursive procedures.

Recursive procedures, particularly when complicated, are better understood dynamically (in terms of their step by step effects) rather than statically (in terms of the static sequence of symbols defining them). The corresponding aspect of the machine language is that the effect of an order is dependent on the very computation which it itself is helping to direct: whether and how often an order is used and to what memory position it refers. All of these are a function of the whole program and the numbers being processed. Thus a program, though a static sequence of symbols, is usually best understood in terms of its dynamic effects, that is, its control of the actual sequential computational process.

To help bridge this gap between the mathematician's description of the desired computation in his own language and the corresponding program in the machine language, von Neumann invented the flow diagram. A flow diagram is a labeled graph composed of enclosures and points connected by lines. The enclosures are of various kinds: operation boxes (specifying non-recursive fragments of the computation as symbolized in the box), alternative boxes (corresponding to conditional transfer of control orders and being labeled with the condition for transfer), substitution and assertion boxes (indicating the values of the indices of the recursions), storage boxes (giving the contents of crucial parts of the memory at certain stages of the computation), and labeled circles representing the beginning and terminus and interconnections. In executing the program corresponding to a given flow diagram, the computer in effect travels through the flow diagram, starting at the beginning circle, executing sequences of orders described in operation boxes, cycling back or branching off to a new part of the diagram according to the criteria stated in alternative boxes, leaving an exit circle in one part of the graph to enter an entrance circle in another part of the graph, and finally stopping at the terminal circle. Direct lines are used to represent the direction of passage through the graph, converging lines meeting at points of the graph. An undirected line is used to connect a storage box to that point of the graph which corresponds to the stage of computation partly described by the contents of the storage box.

It is unnecessary for the programmer to prepare and code a complete flow diagram for a complicated problem. A problem of any considerable complexity is composed of many subproblems, and flow diagrams and subroutines can be prepared for these in advance. It was planned to code subroutines corresponding to a large number of basic algorithms employed in solving problems on a digital computer: decimal-to-binary and binary-to-decimal conversion, double precision arithmetics, various integration and interpolation methods, meshing and sorting algorithms, etc. These subroutines would be available in a library of tapes. To solve a particular problem, the programmer would merely write a "combining routine" which would direct the computer to take the proper subroutines from the tape and modify them appropriately to that particular problem.

The use of combining routines and a library of subroutines was a first step in the direction of using a computer to help prepare programs for itself. Still, in this system, everything written by the programmer must be in the clumsy "machine language." A better procedure is to construct a "programmer's language" in which the programmer will write programs, and then to write a translation program in machine language which directs the machine to translate a program written in the programmer's language into a program stated in machine language. The programming language would be close to the natural and mathematical language normally used by mathematicians, scientists, and engineers, and hence would be easy for the programmer to use. This approach is currently being developed under the name of automatic programming. Von Neumann discussed it under the names "short code" (programmer's language) and "complete code" (machine language).[19]

Von Neumann recognized that the idea of automatic programming is a practical application of Turing's proof that there exists a universal computing machine. A Turing machine is a finite automaton with an indefinitely expandable tape. Any general purpose computer, together with an automatic factory which can augment its tape store without limit, is a Turing machine. Turing's universal computer U has this property: for any Turing machine M there is a finite program P such that machine U, operating under the direction of P, will compute the same results as M. That is, U with P simulates (imitates) M.

Automatic programming also involves simulation. Let U_c be a computer which operates with a machine language inconvenient for the

[19] *The Computer and the Brain*, pp. 70–73.

programmer to use. The programmer uses his more convenient programmer's language. It is theoretically possible to build a machine which will understand the programmer's language directly; call this hypothetical computer M_p. Let P_t be the program (written in the language of machine U_c) which translates from the programmer's language to the machine language of U_c. Then U_c, operating under the direction of P_t, will compute the same results as M_p. That is, U_c with P_t simulates M_p, which is a special case of Turing's universal U with P simulating M.

Note that two languages are employed inside U_c: a machine language which is used directly and a programmer's language which is used indirectly via the translation routine P_t. Von Neumann referred to these as the "primary" and "secondary" language of the machine, respectively. The primary language is the language used for communication and control within the machine, while the secondary language is the language we humans use to communicate with the machine. Von Neumann suggested that by analogy there may be a primary and secondary language in the human nervous system, and that the primary language is very different from any language we know.

> Thus the nervous system appears to be using a radically different system of notation from the ones we are familiar with in ordinary arithmetics and mathematics. . . .
>
> . . . whatever language the central nervous system is using, it is characterized by less logical and arithmetical depth than what we are normally used to.
>
> Thus logics and mathematics in the central nervous system, when viewed as languages, must be structurally essentially different from those languages to which our common experience refers.
>
> . . . when we talk mathematics, we may be discussing a *secondary* language, built on the *primary* language truly used by the central nervous system.[20]

He thought that the primary language of the nervous system was statistical in character. Hence his work on probabilistic logics was relevant to this language. See his discussion of probabilistic logics and reliability in the Third and Fourth Lectures of Part I below and in "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components."

Computer Circuits. From the beginning von Neumann had an interest in the circuits and components of electronic digital computers.

[20] *The Computer and the Brain*, pp. 79–82.

He analyzed the basic physical and chemical properties of matter with the purpose of developing improved computer components.[21] In his lectures on automata theory he compared natural and artificial components with respect to speed, size, reliability, and energy dissipation, and he computed the thermodynamic minimum of energy required for a binary decision. See the Fourth Lecture of Part I of the present volume. His search for physical phenomena and effects that could be used for computing led to his invention of a new component.

This is a subharmonic generator which is driven by an excitation (power) source at frequency nf ($n = 2, 3, 4, \cdots$) and which oscillates at the subharmonic frequency f.[22] The subharmonic generator circuit incorporates an inductance and capacitance circuit tuned to the frequency f. Either the capacitance or inductance is non-linear, and its value varies periodically under the influence of the exciting signal (of frequency nf). The oscillation at frequency f can occur in any of n distinct phases. Each oscillation phase is highly stable when established, but, when the oscillation begins, the choice of phase can easily be controlled by a small input signal of frequency f and of the desired phase. Modulating (turning off and on) the exciting source (of frequency nf) with a square wave (clock signal) of much lower frequency produces alternate passive and active periods, and an input of frequency f can select one of the n phases of oscillation as the exciting signal appears.

To transfer the phase state of one subharmonic generator (a transmitter) to another (a receiver), the transmitter and receiver are coupled through a transformer. The square-wave modulations into transmitter and receiver are of the same frequency but of different phase, so that the transmitter is still on while the receiver is just

[21] Most of his ideas in this area were only discussed with others and never published. A brief reference occurs in *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument, Collected Works* 5.39. Booth, "The Future of Automatic Digital Computers," p. 341, mentions a superconducting storage element discussed with von Neumann in 1947. Von Neumann also did some early work on the MASER. See *Collected Works* 5.420, *Scientific American* (February, 1963) p. 12, and *Scientific American* (April, 1963) pp. 14–15.

[22] "Non-Linear Capacitance of Inductance Switching, Amplifying and Memory Devices." Von Neumann's ideas are also described by Wigington, "A New Concept in Computing."

The parametron, invented independently by E. Goto, embodies essentially the same idea, but is far different in the suggested speed of implementation. See Goto, "The Parametron, a Digital Computing Element which Utilizes Parametric Oscillation." The highest frequencies Goto reports are an exciting frequency ($2f$) of 6×10^6 cycles per second and a clock frequency of 10^5 cycles. According to Wigington, *op. cit.*, von Neumann estimated that an exciting frequency ($2f$) of 5×10^{10} and a clock rate of 10^9 were feasible.

coming on. As a result the receiver begins oscillating at frequency f in phase with the transmitter. The receiver can later transmit its state to another subharmonic generator, and so on down the line. One may use three clock signals, all of the same frequency but of three different phases, and, by exciting interconnected generators with the proper clock signals, transfer information around a system of generators. Each such generator then has an input and an output operating at frequency f, beside the exciting input of frequency nf; the phasing of the two different clock signals to two interconnected generators determines which generator is the receiver and which is the transmitter. The output signal (at f) has much more power than is required for the input signal (at f) to control the phase of the oscillation, and so the subharmonic generator is an amplifier at frequency f, the power for amplification coming from the exciting signal of frequency nf.

Since the oscillation of the subharmonic generator is stable and continues after the subharmonic input from another generator terminates, the device clearly has memory capacity. Switching can also be done with subharmonic generators, in the following way. Let $n = 2$; i.e., let there be two distinct phases of subharmonic oscillation at frequency f, so that the system is binary. Connect the outputs of three transmitting generators to the primary of a transmitter so that the voltages of these outputs add; connect a receiver generator to the secondary of this transformer. The voltage of the transformer secondary will then have the phase of the majority of the transmitting generators, so that the receiving generator will oscillate in this phase. This arrangement realizes a majority element, that is, a three-input switch with delay whose output state is "1" if and only if two or more inputs are in state "1".[23] A negation element may be realized by connecting the output of one generator to the input of another and reversing the direction of the transformer winding. The constants "0" and "1" are realized by sources of the two different phases of the signal of frequency f. The majority element, negation element, and the constant sources "0" and "1" are sufficient to do all computing, so that the central part of a computer can be completely constructed from subharmonic generators.[24]

Von Neumann's Theory of Automata

Introduction. On reviewing the preceding sketch of von Neumann's research accomplishments, one is immediately struck by the tremen-

[23] "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Collected Works* 5.339.

[24] Many computers are so constructed. See Goto, *op. cit.*

dous combination of breadth and depth revealed in those accomplishments. Particularly notable is the extent to which von Neumann's achievements range from the purely theoretical to the extremely practical. It should be added in the latter connection that he was among the first to recognize and promote the tremendous potentialities in computers for technological revolution and the prediction and control of man's environment, such as the weather.

Von Neumann was able to make substantial contributions to so many different fields because he possessed a rare combination of different abilities along with wide interests. His quick understanding and powerful memory enabled him to absorb, organize, retain, and use large quantities of information. His wide interests led him to work in and keep contact with many areas. He was a virtuoso at solving difficult problems of all kinds and at analyzing his way to the essence of any situation.

This wide range of interests and abilities was one of von Neumann's great strengths as a mathematician and made him an applied mathematician par excellence. He was familiar with the actual problems of the natural and engineering sciences, on the one hand, and the abstract methods of pure mathematics on the other. He was rare among mathematicians in his ability to communicate with scientists and engineers. This combination of theory and practice was deliberately cultivated by von Neumann. He was a careful student of the history and nature of scientific method and its relation to pure mathematics[25] and believed that mathematics must get its inspiration from the empirical sciences.

Given his background and type of mind, it was natural for von Neumann to begin to construct a general theory of computers. Being aware of the important similarities between computers and natural organisms, and of the heuristic advantages in comparing such different but related systems, he sought a theory that would cover them both. He called his proposed systematic theory the "theory of automata." This theory of automata was to be a coherent body of concepts and principles concerning the structure and organization of both natural and artificial systems, the role of language and information in such systems, and the programming and control of such systems. Von Neumann discussed the general nature of automata theory at several places in Part I and in Chapter 1 of Part II of the present volume.

Von Neumann's early work on computer design and programming

[25] See Chapter 1 of *Theory of Games and Economic Behavior;* "The Mathematician," *Collected Works* 1.1–9; and "Method in the Physical Sciences," *Collected Works* 6.491–498.

led him to recognize that mathematical logic would play a strong role in the new theory of automata. But for reasons to be mentioned later, he thought that mathematical logic in its present form, though useful in treating automata, is not adequate to serve as "the" logic of automata. Instead, he believed that a new logic of automata will arise which will strongly resemble and interconnect with probability theory, thermodynamics, and information theory. It is obvious from all this that von Neumann's theory of automata will, in the beginning at least, be highly interdisciplinary.

Unfortunately, because of his premature death, von Neumann was unable to put in final form any of the research he was doing in automata theory. In his last work on this subject he said that "it would be very satisfactory if one could talk about a 'theory' of such automata. Regrettably, what at this moment exists . . . can as yet be described only as an imperfectly articulated and hardly formalized 'body of experience'." [26] Von Neumann's accomplishments in this area were nevertheless substantial. He outlined the general nature of automata theory: its structure, its materials, some of its problems, some of its applications, and the form of its mathematics. He began a comparative study of artificial and natural automata. Finally, he formulated and partially answered two basic questions of automata theory: How can reliable systems be constructed from unreliable components? What kind of logical organization is sufficient for an automaton to be able to reproduce itself? The first of these questions is discussed in his "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." The second question is discussed in the Fifth Lecture of Part I and in Part II of the present volume.

I do not know how von Neumann was led to these two problems, but on the basis of his interests and what he has written it is plausible that they arose out of his actual work with computers in the following way. The new electronic computers were revolutionary because in contrast to earlier computing systems (humans, mechanical and electromechanical machines, and combinations thereof) they could do large quantities of computation automatically and rapidly. The advances through the ENIAC, the proposed EDVAC, and the Institute for Advanced Study computer, were all big steps in the direction of more powerful computers. His interest in solving non-linear partial differential equations in general, and in the equations for predicting the weather in particular, would naturally lead him to desire ever more powerful machines and to look for and try to remove the basic limitations blocking the construction of such machines. As a consultant for

[26] *The Computer and the Brain*, p. 2.

government and industry he was very influential in promoting the design and construction of larger computers.

Von Neumann compared the best computers that could be built at the time with the most intelligent natural organisms and concluded that there were three fundamental factors limiting the engineer's ability to build really powerful computers: the size of the available components, the reliability of these components, and a lack of a theory of the logical organization of complicated systems of computing elements. Von Neumann's work on componentry was directed toward the first limitation, and his results on reliability and self-reproduction each contribute toward removing both the second and the third limitations. In his "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," he gave two methods of overcoming the unreliability of the components, not by making them more reliable, but by organizing them so that the reliability of the whole computer is greater than the reliability of its parts. He regarded his work on probabilistic logics as a step in the direction of the new logic of automata. His work on self-reproduction also belongs to the theory of complicated automata. He felt that there are qualitatively new principles involved in systems of great complexity and searched for these principles in the phenomenon of self-reproduction, which clearly depends on complexity. It is also to be expected that because of the close relation of self-reproduction to self-repair, results on self-reproduction would help solve the reliability problem.

Thus von Neumann was especially interested in complex automata; he wanted a theory of the logical organization of complicated systems of computing elements. His questions about reliability and self-reproduction are particularly germane to complex automata.

Two further points are relevant. First, von Neumann believed that in starting a new science one should begin with problems that can be described clearly, even though they concern everyday phenomena and lead to well known results, for the rigorous theory developed to explain these phenomena can provide a base for further advances.[27] His problems of reliability and self-reproduction are of this kind. Second, von Neumann believed that the lack of an adequate theory of complicated automata is an important practical barrier to building more powerful machines. He explicitly stated that until an adequate theory of automata exists there is a limit in the complexity and capacity of the automata we can fabricate.[28]

[27] *Theory of Games and Economic Behavior*, Secs. 1.3 and 1.4.

[28] "The General and Logical Theory of Automata," *Collected Works* 5.302–306.

Natural and Artificial Automata. The scope of the theory of automata and its interdisciplinary character are revealed by a consideration of the two main types of automata: the artificial and the natural. Analog and digital computers are the most important kinds of artificial automata, but other man-made systems for the communication and processing of information are also included, for example, telephone and radio systems. Natural automata include nervous systems, self-reproductive and self-repairing systems, and the evolutionary and adaptive aspects of organisms.

Automata theory clearly overlaps communications and control engineering on the one hand, and biology on the other. In fact, artificial and natural automata are so broadly defined that one can legitimately wonder what keeps automata theory from embracing both these subjects. Von Neumann never discussed this question, but there are limits to automata theory implicit in what he said. Automata theory differs from both subjects in the central role played by mathematical logic and digital computers. Though it has important engineering applications, it itself is a theoretical discipline rather than a practical one. Finally, automata theory differs from the biological sciences in its concentration on problems of organization, structure, language, information, and control.

Automata theory seeks general principles of organization, structure, language, information, and control. Many of these principles are applicable to both natural and artificial systems, and so a comparative study of these two types of automata is a good starting point. Their similarities and differences should be described and explained. Mathematical principles applicable to both types of automata should be developed. Thus truth-functional logic and delay logic apply to both computer components and neurons, as does von Neumann's probabilistic logic. See the Second and Third Lectures of Part I of the present volume. Similarly, von Neumann's logical design of a self-reproducing cellular automaton provides a connecting link between natural organisms and digital computers. There is a striking analogy with the theory of games at this point. Economic systems are natural; games are artificial. The theory of games contains the mathematics common to both economic systems and games,[29] just as automata theory contains the mathematics common to both natural and artificial automata.

Von Neumann himself devoted considerable attention to the com-

[29] *Theory of Games and Economic Behavior,* Secs. 1.1.2 and 4.1.3.

parison of natural and artificial automata.[30] Scientific knowledge of the automata aspects of natural organisms has advanced very rapidly in recent years, and so there is now a much more detailed basis for the comparison than at the time von Neumann wrote, but his general approach and conclusions are nevertheless of interest. We will outline his reflections under the following headings: (1) The analog-digital distinction, (2) the physical and biological materials used for components, (3) complexity, (4) logical organization, and (5) reliability.

(1) Von Neumann discussed the analog-digital distinction at length and found it to be an illuminating guide in his examination of natural automata. See the First and Fourth Lectures of Part I. His most general conclusion was that natural organisms are mixed systems, involving both analog and digital processes. There are many examples, of which two will suffice here. Truth-functional logic is applicable to neurons as a first approximation, but such neural phenomena as refraction and spatial summation are continuous rather than discrete. In complicated organisms digital operations often alternate with analog processes. For example, the genes are digital, while the enzymes they control function analogically. Influenced by his knowledge of natural automata von Neumann proposed a combined analog-digital computing scheme.[31] This is a good example of the effect of the study of natural systems on the design of artificial ones.

(2) Von Neumann compared the components in existing natural and artificial automata with respect to size, speed, energy requirements, and reliability, and he related these differences to such factors as the stability of materials and the organization of automata. Computer components are much larger and require greater energy than neurons, though this is compensated for in part by their much greater speed. These differences influence the organization of the system: natural automata are more parallel in operation, digital computers more serial. Part of the difference in size between a vacuum tube and a neuron can be accounted for in terms of the mechanical stability of the materials used. It is relatively easy to injure a vacuum tube and difficult to repair it. In contrast, the neuron membrane when injured is able to restore itself. Von Neumann calculated the thermodynamical minimum of energy that must be dissipated by a computing element and concluded that in theory computing elements could be of the

[30] Norbert Wiener also made valuable comparisons of natural and artificial systems in his *Cybernetics*, though in a somewhat different way. The two men were aware of each other's work—see *Cybernetics* (particularly the "Introduction") and von Neumann's review of it.

[31] Sec. 12 of "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Collected Works* 5.372–377.

order of 10^{10} times more efficient in the use of energy than neurons. See the Fourth Lecture of Part I. His comparison of natural and artificial components no doubt influenced his work on computer components.

(3) Man is, inter alia, a natural automaton which is obviously very much more complex than any artificial automaton he has so far constructed. Because of this complexity he understands the details of his own logical design much less than that of the largest computer he has built. Von Neumann thought that the chief problems of automata theory center around the concept of complexity. This very concept needs rigorous definition. Automata theory should relate the logical organization of complex automata to their behavior. A theory which did this would enable us to develop the logical design of artificial automata capable of carrying out some of the most difficult and advanced functions performed by humans as well as many other complex functions that humans cannot perform, such as solving large systems of non-linear partial differential equations. The problem of reliability is especially crucial in complex systems. Von Neumann speculated that extremely complex systems involve new principles. He thought, for example, that below a certain level, complexity is degenerative, and self-reproduction is impossible. He suggested that, generally speaking, in the case of simple automata a symbolic description of the behavior of an automaton is simpler than the automaton itself, but that in the case of exceedingly complex automata the automaton is simpler than a symbolic description of its behavior. See the Second Lecture of Part I.

(4) In discussing the relative speeds of natural and artificial components we noted that natural automata tend to be more parallel in operation and artificial automata tend to be more serial in operation. When planning an automaton or a computation, one can choose somewhat the extent to which it is parallel or serial, but there are definite limits to this—e.g., in a serial computation a later operation may depend on an earlier one and hence cannot proceed simultaneously with it. Moreover, this choice affects other aspects of the machine, particularly the memory requirements, for a datum that is to be operated on later must be stored until it is needed. The memory of an artificial automaton is generally organized in a hierarchy, different levels of the hierarchy operating at different speeds. In a typical computer there are high-speed electronic registers, slower speed magnetic cores, and much slower magnetic tape units. In addition there is the wiring of the machine itself, which provides the unalterable organization of the system. Von Neumann discussed machine memory hier-

archies, and said that we should look for similar hierarchies in natural automata. Pulses circulating in neuron cycles, the change of neural thresholds with use, the organization of the nervous system, and the coding of the genes together constitute such a hierarchy.

The organization of an automaton is to be distinguished from the organization of a particular computation in that automaton. When both are taken into account, the difference between natural and artificial automata with respect to serial vs. parallel operation seems to be accentuated. Von Neumann spoke in this connection of the "logical depth" of a computation.[32] A computation consists of a large number of basic logical steps (switching and delay), the result of each step depending on certain prior steps. We will call any sequence of steps, each of which depends critically on its predecessor, a "calculation chain." The logical depth of a computation is the number of logical steps in its longest calculation chain. Because of their great speed, digital computers are used to perform computations of exceedingly great logical depth. For the final answer to be useful its error must be kept small, and this results in a very strong reliability requirement on each logical step. This brings us to von Neumann's fifth and last main point of comparison between natural and artificial automata.

(5) The first electronic digital computers had little equipment for the automatic detection of failure. They were designed and wired with extreme care and were constructed of components especially selected for great reliability. Programs were written with care and laboriously checked. Diagnostic programs were used to detect machine errors, and various procedures (e.g., differencing) were employed to check the computed results. Thus these machines were designed, built, and used in such a way that, hopefully, a single malfunction would be noted before a second occurred. The machine would then be stopped and the fault isolated by an analytic procedure. As von Neumann pointed out in the Fourth Lecture of Part I, this method of handling errors would obviously not be satisfactory for extremely complicated automata. The very design and construction of such large automata would result in many mistakes. Moreover, the large number of components would result in a very short mean free path between errors and make localization of failures too difficult. Natural automata are clearly superior to artificial ones in this regard, for they have strong powers of self-diagnosis and self-repair. For example, the human brain can suffer great damage from mechanical injury or disease and still continue to

[32] *The Computer and the Brain*, pp. 27, 79. He also spoke of the logical depth of a language. See *ibid.*, pp. 81–82 and the discussion of the primary language of the nervous system at p. 15 above.

function remarkably well. Natural and artificial automata are thus organized in very different ways for protection against errors. Von Neumann's work on reliability serves to link these two types of automata in this respect.

Mathematics of Automata Theory. Von Neumann intended the theory of automata to be highly mathematical and logical. The study of actual automata, both natural and artificial, and of their operation and interaction, provides the empirical source of this formal component of automata theory. This is in keeping with von Neumann's belief that mathematics derives inspiration and ideas from empirical subject matter.

The close connection between mathematical logic and automata was well known to von Neumann when he wrote on automata theory. Kurt Gödel had reduced mathematical logic to computation theory by showing that the fundamental notions of logic (such as well-formed formula, axiom, rule of inference, proof) are essentially recursive (effective).[33] Recursive functions are those functions which can be computed on Turing machines, and so mathematical logic may be treated from the point of view of automata.[34] Conversely, mathematical logic may be applied to the analysis and synthesis of automata. The logical organization of an automaton can be represented by a structure of idealized switch-delay elements and then translated into logical symbolism. See the Second Lecture of Part I.

Because of the intimate connection between automata and logic, logic will be at the heart of the mathematics of automata theory. Indeed, von Neumann often spoke of a "logical theory of automata" rather than merely a "theory of automata." Nevertheless, he felt that the mathematics of automata theory would also have some formal characteristics very different from those of logic. Roughly speaking, mathematics can be divided into the discrete and the continuous. Logic is a branch of discrete mathematics and is highly combinatorial. Von Neumann thought that automata mathematics should be closer to the continuous and should draw heavily on analysis. He thought that the specific problems of automata theory require this, and he felt that there is a general advantage in an analytical as opposed to a combinatorial approach in mathematics.

There is an important topic in the theory of automata that requires

[33] "Uber formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I." The notion of theoremhood is not in general recursive, but the theorems of a formal language are always recursively enumerable.

[34] Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem" and "Computability and λ-Definability."

a more analytical treatment than is usual in logic. Automata theory must encompass the probability of failure of a component. Mathematical logic treats only the perfect or deterministic operation of idealized switch-delay elements; it provides no theoretical treatment of error. Hence in using mathematical logic for actual design one must supplement it by considerations that lie outside the subject itself. Von Neumann wanted a probabilistic logic which would handle component malfunction as an essential and integral part of automata operation. While probability theory is strongly combinatorial, it also makes important contacts with analysis.

Including the probability of failure in the logic of automata forces one to consider the size of a computation. The usual approach in mathematical logic is to consider whether or not something can be accomplished by an automaton in a finite number of steps, regardless of how large the number is. But on any realistic assumption about component failure, the larger the calculation the more likely the machine will err during it, and the less likely the result will be correct. This concern for the size of a computation also arises from our practical interests in automata. Computers are built in order to produce certain results in the available time. Since many of the functions we desire computers to perform are now performed by humans, it should be kept in mind in this connection that man is a finite automaton, not a Turing machine. Von Neumann did not suggest how to construct a theory of the sizes of computations. Presumably this theory would be based on a *quantitative* notion of "amount of computation" which would take into account both the length of a calculation (the "logical depth" of p. 24 above) and its width (the amount of parallelism in it).

Thus a theory of the quantity of computation and the likelihood of its being wrong must include continuous as well as discrete mathematics.

All of this will lead to theories which are much less rigidly of an all-or-none nature than past and present formal logic. They will be of a much less combinatorial, and much more analytical, character. In fact, there are numerous indications to make us believe that this new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information. Its techniques are indeed much more analytical than combinatorial, which again illustrates the point that I have been trying to make above.[35]

[35] "The General and Logical Theory of Automata," *Collected Works* 5.304. The next quotation is from the same article, 5.303.

Von Neumann also held that there is a methodological advantage in employing analysis in the mathematics of automata.

> Everybody who has worked in formal logic will confirm that it is one of the technically most refractory parts of mathematics. The reason for this is that it deals with rigid, all-or-none concepts, and has very little contact with the continuous concept of the real or of the complex number, that is, with mathematical analysis. Yet analysis is the technically most successful and best-elaborated part of mathematics. Thus formal logic is, by the nature of its approach, cut off from the best cultivated portions of mathematics, and forced onto the most difficult part of the mathematical terrain, into combinatorics.

This comment is particularly significant since von Neumann made important contributions to discrete mathematics. In *Theory of Games and Economic Behavior* he stated that the mathematics to be developed for social theory should emphasize combinatorics and set theory rather than differential equations.[36]

In his own work in automata theory von Neumann moved from the discrete toward the continuous. His probabilistic logic is an example. After presenting this logic, he proposed a mixed analog-digital computing system closely related to it.[37] His first models of self-reproduction were discrete, but he hoped later to develop a continuous model of self-reproduction. See Section 1.1.2.3 of Part II of the present volume.

We noted before that von Neumann often referred to his theory of automata as a "logical theory of automata." He also called it "theory of automata and information" and sometimes just "theory of information," indicating the strong role that he expected information theory to play in the subject. He divided the theory of control and information into two parts: a strict part and a probabilistic part. The rigorous or strict part includes mathematical logic as extended to cover finite automata and Turing machines. The statistical or probabilistic part includes the work of Shannon on information theory[38] and von Neumann's probabilistic logic. Von Neumann regarded his probabilistic logic as an extension of rigorous logic.

There is a close connection between information theory and thermodynamics, both subjects employing the concept of probability in very much the same way. See the Third Lecture of Part I, especially the quotation from von Neumann's review of Wiener's *Cybernetics*.

[36] Sec. 4.8.3. Cf. Sec. 1.2.5.

[37] Sec. 12 of "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Collected Works* 5.372–377.

[38] "A Mathematical Theory of Communication."

Von Neumann mentioned two further connections between thermodynamics and automata theory. First, he found an analog of thermodynamic degeneration in the theory of self-reproducing automata: below a certain minimum level, complexity and degree of organization are degenerative, but above that level they are not degenerative and may even increase. Second, he discussed the thermodynamic aspect of the concept of balance in computing machine design. The efficiency of a computer depends on the proper balance of its different parts with respect to speed and size. For example, in the memory hierarchy the different kinds of memory (e.g., transistor, core, tape) should be matched to one another in size and speed. A computer in which the arithmetic unit is too fast for the memory, or the memory is too small, is like a heat engine which is inefficient because large temperature differences exist between two parts of it. The efficiency of a computer must be defined relative to its environment (i.e., the problems it is to solve), just as the efficiency of a heat engine depends on its environment. These problems of balance and matching are handled empirically by engineers. Von Neumann wanted a quantitative theory of balance akin to thermodynamics.

To conclude, von Neumann thought that the mathematics of automata theory should start with mathematical logic and move toward analysis, probability theory, and thermodynamics. When it is developed, the theory of automata will enable us to understand automata of great complexity, in particular, the human nervous system. Mathematical reasoning is performed by the human nervous system, and the "primary" language in which mathematical reasoning takes place is analogous to the primary language of a computing machine (p. 15 above). It is thus quite possible that automata theory will affect logic and our fundamental concepts of mathematics.

> I suspect that a deeper mathematical study of the nervous system . . . will affect our understanding of the aspects of mathematics itself that are involved. In fact, it may alter the way in which we look on mathematics and logics proper.[39]

Now logic lies at the foundation of mathematics; therefore, if von Neumann's suggestion is true, automata theory will move full circle: starting at the foundation of mathematics and ending there.

ARTHUR W. BURKS

[39] *The Computer and the Brain*, p. 2; cf. pp. 70–82. See also Ulam, "John von Neumann, 1903–1957," p. 12.

PART *One*

Theory and Organization of Complicated Automata

EDITORIAL NOTE

[The editor's writing is in brackets. The reconstructed edition of von Neumann's work is not bracketed, but much of the unbracketed text is heavily edited. See the Preface.]

First Lecture

COMPUTING MACHINES IN GENERAL

Conceptual and numerical methods in mathematics. The role of the latter in applied mathematics and in mathematical physics. Their role in pure mathematics. The situation in analysis. Numerical procedures as heuristic tools.

Various forms of the numerical approach: Analog and digital.

The analog procedure: The use of the physical experiment as a substitute for computing. Analog computing machines.

The digital procedure: Manual computing. Simple machines. Fully automatic computing.

The present status of computing machines. Present roles of analog and digital machines. Questions of speed, programming, and precision.

The concept of an elementary operation in a computing machine. Its role in analog machines and in digital machines. Observations on analog componentry. Observations on digital componentry.

The relay organ. Main forms: The electro-mechanical relay. The vacuum tube. Other possible relay organs.

Measurement of the length or complexity of a numerical calculation. Logical and arithmetical operations. Linear and non-linear arithmetical operations. The role of the number of multiplications. Stability of the statistical characteristics of various parts of mathematics. The special role of analysis.

Various characteristic levels of length or complexity. Characteristic problem lengths for automatic digital machines.

Precision requirements.

Memory requirements: Measurement of memory capacity. The decisive characteristics of a memory: Access time and capacity. Reasons for a hierarchic organization of memory. Actual memory requirements of an automatic digital machine.

Input-output: Main available media.

The concept of balance: Speed balance of various components. Balance between memory capacity in various stages of the hierarchy and speeds. Balance between speed and precision. Balance between speed, memory capacity, and programming capacity.

Thermodynamical aspects of the concept of balance. Thermodynamical aspects of the memory capacity. Need for a quantitative theory, contrasted

with the present empirical procedures. Preliminary remarks on reliability and errors.

Ladies and gentlemen, I wish to thank you for your very friendly welcome on my five occasions to talk to you, and I hope that I will be able to offer something for the variety of interests which are represented here. I will talk about automata—the behavior of very complicated automata and the very specific difficulties caused by high complication. I shall discuss briefly the very plausible, very obvious analogies which come to mind between artificial automata and organisms, which within a certain limit of their functioning are natural automata. We must consider the similarities, the dissimilarities, the extent to which the dissimilarities are due to our skill or clumsiness (the latter being the more normal phenomenon), and the extent to which these dissimilarities are really matters of principle.

Today I will talk chiefly about artificial automata, and specifically about one variety of artificial automata, namely, computing machines. I will talk about their role in the near past and present and about what to expect from them in the future.

I am talking about computing machines partly because my interests in the subject of automata are mathematical and, from the mathematical point of view, computing machines are the most interesting and most critical automata. But quite apart from this ex parte argument from the mathematical side, there is the important question of automata of very, very high complexity. Of all automata of high complexity, computing machines are the ones which we have the best chance of understanding. In the case of computing machines the complications can be very high, and yet they pertain to an object which is primarily mathematical and which we understand better than we understand most natural objects. Therefore, by considering computing machines, we can discuss what we know and what we do not know, what is right and what is wrong, and what the limitations are, much more clearly than if we discussed other types of automata. You will see that our discussion of complex automata is very far from perfect and that one of our main conclusions is that we need very badly a theory which we do not at this moment possess.

Let me first say something from the properly mathematical side, namely, the role which computing machinery has played or might play in mathematics and in adjacent subjects. Speaking of numerical computing in general, it is not necessary to discuss what role it can play in many applications of mathematical methods. It's perfectly clear that numerical computing plays a large role in engineering. If more

computing and faster computing could be done, one would have even more uses for computing in engineering.

Let me come now to the less obvious things. In physics, particularly in theoretical physics, it is clear that mathematical methods play a great role and that to a large extent they are of the same variety as pure mathematics, that is, they are abstract and analytical. However, effective computing plays a role in physics which is larger than the role one would expect it to have in mathematics proper. For instance, there are large areas of modern quantum theory in which effective iterative computing could play a large role. A considerable segment of chemistry could be moved from the laboratory field into the purely theoretical and mathematical field if one could integrate the applicable equations of quantum theory. Quantum mechanics and chemistry offer a continuous spectrum of problems of increasing difficulty and increasing complexity, treating, for example, atoms with increasing numbers of electrons and molecules with increasing numbers of valence electrons. Almost any improvement in our standards of computing would open important new areas of application and would make new areas of chemistry accessible to strictly theoretical methods.

However, I will not go into great detail on this subject either but would like to give you a brief indication of what role this kind of computing might play in mathematics proper, that is, in pure mathematics. In pure mathematics the really powerful methods are only effective when one already has some intuitive connection with the subject, when one already has, before a proof has been carried out, some intuitive insight, some expectation which, in a majority of cases, proves to be right. In this case one is already ahead of the game and suspects the direction in which the result lies. A very great difficulty in any new kind of mathematics is that there is a vicious circle: you are at a terrible disadvantage in applying the proper pure mathematical methods unless you already have a reasonably intuitive heuristic relation to the subject and unless you have had some substantive mathematical successes in it already. In the early stages of any discipline this is an enormous difficulty; progress has an autocatalytic feature. This difficulty may be overcome by some exceptionally lucky or exceptionally ingenious performance, but there are several outstanding instances where this has failed to happen for two, three, or four generations.

One of these areas which has been conspicuous for some time is the area of non-linear problems. The great successes of the nineteenth century, as well as of modern analysis, were in linear problems. We have much less experience with non-linear problems, and we can say

practically nothing about the majority of non-linear partial differential equations. We have never been successful with these at all, and therefore we have absolutely no idea what the difficulties are.

In those few domains where some progress had been made it was usually for different reasons, for instance, because some very usual physical phenomenon was tied up with the mathematical problems and therefore one had a non-mathematical, physical approach. In these domains scientists discovered the most surprising types of singularities, which have absolutely no analogs in the linear domain we know so well, that is, absolutely no analogs in those parts of mathematical analysis like complex number theory, and so on. These experiences make a fairly convincing case that completely new methods will be needed for non-linear problems. The classical example for this is a non-linear partial differential equation for compressible, non-viscous flow, which led to the discovery of the phenomenon of shocks. In a problem in which it seemed that only continuous solutions should exist, discontinuous solutions suddenly play an enormous role, and without proper regard for these one cannot prove the uniqueness or the existence of solutions. Furthermore, these irregular solutions behave in a very peculiar manner and violate a number of the regularities which we had reason to believe, from other forms of analysis, were well established.

Another good example is the phenomenon of turbulence in the viscous case, where one suddenly discovers that the really important solutions to a problem which has very high symmetry do not possess that symmetry. From a heuristic point of view, the important thing is not to find the simplest solution of the problem, but rather to analyze statistically certain large families of solutions which have nothing in common with each other except certain statistical traits. These prevalent statistical traits are the real roots of the problem and cause very peculiar singularities in many individual solutions. In all these cases there is reason to believe that we will have great difficulty in making analytical progress. The problem of turbulence has been around for 60 years, and analytical progress in solving it has been very small.[1]

Almost all of the correct mathematical surmises in this area have come in a very hybrid manner from experimentation. If one could calculate solutions in certain critical situations like those we have mentioned, one would probably get much better heuristic ideas. I

[1] [See further von Neumann's *Collected Works* 5.2–5 and Birkhoff's *Hydrodynamics*.]

will try to give some indications of this later, but I wanted to point out that there are large areas in pure mathematics where we are blocked by a peculiar inter-relation of rigor and intuitive insight, each of which is needed for the other, and where the unmathematical process of experimentation with physical problems has produced almost the only progress which has been made. Computing, which is not too mathematical either in the traditional sense but is still closer to the central area of mathematics than this sort of experimentation is, might be a more flexible and more adequate tool in these areas than experimentation.

Let me come to the subject proper and first say a few things about the general traits of computing processes and computing machines. As you probably know, the main types of computing machines existing or being discussed or planned at this moment fall into two large classes: super-analog devices and digital devices. Let me first describe the analog devices or the wider class, inasmuch as a proper definition is usually given for the digital class, and analogs are essentially everything else.

Roughly speaking, an analog calculation is one in which you look at some physical process which happens to have the same mathematical equations as the process you're interested in, and you investigate this physical process physically. You do not take the physical process which you are interested in, because that is your whole reason to calculate. You always look for something which is like it but not exactly the same thing.

The smallest modification you may make is to use a different scale, which is possible in certain problems. A slightly larger modification is to use a different scale and also change certain things which are not exactly scales. For instance, when you try an aerodynamical experiment in a wind tunnel you scale it, but you scale not only the linear dimensions but also the velocity of sound. The only way to scale the velocity of sound is to go to a lower temperature, and there you really need insight. You must know that the phenomenon you're concerned with does not depend on temperature. You then discover that it is easier to try it at a lower temperature and with much smaller dimensions than to carry out the actual process you are interested in. In this way a wind tunnel for aerodynamical experimentation is in a sense an analog computing device. This is not a completely fair comparison because a wind tunnel does a good deal beside computing, but still in a large range of application (which is certainly not much less than 50 per cent) it is just an analog computing device.

You come very quickly then to cases in which you will not do

exactly this because it is not possible or not convenient to find a physical process which has exactly the same equations as the problem you are interested in. But you may still find, for example, three different processes which have the same equations as three parts of the problem and which can be aligned in such a manner that if you perform one after the other you get the complete answer. From this there is a continuous transition to situations where you actually break up the problem mathematically into the elementary operations of arithmetic: multiplication, addition, subtraction, and division.

[Von Neumann next discussed physical analog processes for adding, multiplying, subtracting, and dividing. He covered both electrical and mechanical analog processes, and the way numbers are represented in each. He said, "What passes in any axiomatic treatment of mathematics for the elementary operations of arithmetic, the four species of arithmetical addition, subtraction, etc., need not be the elementary operations of a computing machine, specifically of an analog computing machine." He explained how a differential analyzer multiplies two constants by integrating and subtracting. See *The Computer and the Brain 3–5*, *Collected Works* 5.293.]

[Von Neumann then took up digital machines. He remarked that in the last 10 years purely digital devices had become relatively much more important than analog devices. He discussed the components of digital machines (toothed wheels, electromechanical relays, vacuum tubes, and nerve cells), the speeds of these components (including both response time and recovery time), and the need for power amplification in these components. He stressed the role of the basic logical operations (such as sensing a coincidence) in control mechanisms, including "the most elaborate control mechanism known, namely, the human nervous system." See *The Computer and the Brain* 7–10, 30, 39–47. He next turned to the problem of measuring the complexity of automata.]

It is not completely obvious how to measure the complexity of an automaton. For computing machines, probably the reasonable way is to count how many vacuum tubes are involved. This is somewhat ambiguous, because certain current types of vacuum tubes are in reality two vacuum tubes inside one envelope, in which case one is never quite sure which one of the two he is talking about. Another reason is that a great deal enters into computing machine circuitry aside from vacuum tubes: electrical equipment like resistors, capacitances, and possibly inductances. Nevertheless, the ratio of these to the vacuum tubes is tolerably constant, and therefore the number of tubes is probably a reasonable measure of complexity.

The largest calculating machine ever used to date contains twenty thousand vacuum tubes.[2] Now the design of this machine is very different from what any vacuum tube machine of the future is likely to be like, and so this machine is not quite typical. The computing machines which most people are thinking about as the computing machines of the immediate future are to be smaller than this, probably having 2 to 5 thousand tubes. So, roughly speaking, the order of magnitude of the complexity of these machines is 10 thousand.

To give you a comparison with natural organisms, the number of nerve cells in a natural organism can be very different from this. The number of nerve cells in the human central nervous system has been estimated to be 10 billion. This number is so large that of course one has absolutely no experience with such orders of magnitude. It's terribly difficult to form any reasonable guess as to whether things which are as complex as the behavior of a human being can or cannot be administered by 10 billion switching organs. No one knows exactly what a human being is doing, and nobody has seen a switching organ of 10 billion units; therefore one would be comparing two unknown objects.

Let me say a few things which relate more specifically to computing machines. If you can repeat an elementary act like switching with a vacuum tube 1 million times per second, that does not mean of course that you will perform anything that is mathematically relevant 1 million times per second. In estimating how fast a computing machine can operate, there are all kinds of standards. There's a reasonable agreement that one ought to count the number of multiplications performed in a second. By multiplications I mean the multiplication of two full sized numbers with the precision with which the machine is running. There is good reason to believe that the precision with which these things ought to run is of the order of 10, 12, or 14 decimal digits. A machine of reasonable design in which the elements have a speed of about 1 million per second will probably multiply somewhere in the neighborhood of 1 millisecond.

No matter how you organize a computing machine, you simply cannot count on using it to get 100 per cent efficiency. By that I mean that it's impossible, with our present information on the subject, to organize the machine in such a manner that a multiplier which can multiply in one thousandth of a second will really be fed the necessary

[2] [This is the ENIAC, which is described in Burks, "Electronic Computing Circuits of the ENIAC" and "Super Electronic Computing Machine," Goldstine and Goldstine, "The Electronic Numerical Integrator and Computer (ENIAC)," and Brainerd and Sharpless, "The ENIAC."]

data for multiplication. There is a good deal else to be done, namely, making up your mind what numbers you want, getting the numbers, disposing of the result, deciding whether you should do the same thing once again or whether you should do something else, and so on. This is remarkably similar to paper pushing and adding on paper directly, except that what you're pushing is not on paper.

From a logical point of view the efficiency is probably of the order of 1 in 10 or a little better. By that I mean that in any reasonable, logical description of what you are doing, in a code which corresponds to prevalent procedures in formal logics, somewhere between a fifth and a tenth of the orders will be orders to multiply. Since multiplication is somewhat slower than the other operations, in what most people think is a well integrated, well balanced machine, you will probably spend something like one quarter to one half of the time multiplying. So, if you have a multiplier which can effect a multiplication in 1 millisecond, you are doing fairly well to get 500 multiplications per second.

In human computing aided by a desk machine the same number will be perhaps, 2 multiplications per minute. So the discrepancy, the acceleration factor could probably be pushed to 100 thousand or something like that. But to get out of this range we'll probably have to depart from present techniques quite radically.

From the mathematical point of view the question arises whether anything could be done with this speed if one had it. I would like to point out very emphatically that there are very good reasons for asking for anything the traffic will bear, for this speed, 10 times more, a hundred times, a thousand times, or a million times. Problems there are good reasons to solve would justify a great deal more speed than anyone can think of at this moment. [Von Neumann gave as examples quantum mechanical calculations on atomic and molecular wave functions (where the combinatorial difficulties go up very fast as the number of electrons goes up), and the problem of turbulence.]

Although it doesn't belong strictly to the subject, let me point out that we will probably not want to produce vast amounts of numerical material with computing machines, for example, enormous tables of functions. The reason for using a fast computing machine is not that you want to produce a lot of information. After all, the mere fact that you want some information means that you somehow imagine that you can absorb it, and, therefore, wherever there may be bottlenecks in the automatic arrangement which produces and processes this information, there is a worse bottleneck at the human intellect into which the information ultimately seeps.

The really difficult problems are of such a nature that the number of data which enter is quite small. All you may want to know is a few numbers, which give a rough curve, or one number. All you may want in fact is a "yes" or a "no," the answer as to whether something is or is not stable, or whether turbulence has or has not set in. The point is that you may not be able to get from an input of, say, 80 numbers to an output of 20 numbers without having, in the process, produced a few billion numbers in which nobody is interested. But the process is such that the volume of numerical material handled first expands and then contracts again, and, while it starts on a low level, say with 100 numbers, and ends on a low level, say with 10 numbers, its maximum in between is large, say a few thousand, and the number of successive generations is large, so that you have handled 10 billion numbers before you are through. These figures are quite realistic; it would be easy to find problems which have about this numerical makeup.

You may have noticed that I have already introduced one distinction, namely, the total numerical material produced in a process. The other thing which matters is how much you need simultaneously. This is probably the most vexing problem in modern computing machine technology. It's also quite a problem from the point of view of the human organism, namely, the problem of memory. You see, all these automata really consist of two important parts: the general switching part (an active part which affects the logical operations the automaton is supposed to perform), and the memory (which stores information, chiefly intermediate results which are needed for a while and are then discarded and replaced by others).

In computing machines, the methods to do the active part, the arithmetical and control circuits, have been well known for years. The memory questions were much more critical and much more open throughout the last decade, and are even more critical and more open now. In the human organism, we know that the switching part is composed of nerve cells, and we know a certain amount about their functioning. As to the memory organs, we haven't the faintest idea where or what they are. We know that the memory requirements of the human organism are large, but on the basis of any experience that one has with the subject, it's not likely that the memory sits in the nervous system, and it's really very obscure what sort of thing it is.[3] Hence in both the computer and the human nervous system, the dynamic part (the switching part) of the automaton is simpler than the memory.

[3] [This point is discussed further in *The Computer and the Brain* 63–69.]

[Von Neumann next discussed how to measure memory capacity. He suggested using the logarithm (to the base two) of the configuration number (i.e., the number of alternatives). See *Collected Works* 5.341–342. He then estimated the memory capacity of an ordinary printed page to be about 20 thousand units, and remarked that this is about the memory capacity of the digital computers under consideration at that time.]

This shows where the real price of speed lies. A large modern computing machine is a very expensive object, an object which it takes a long time to build and which is a very tricky thing after you've got it. Yet it is supposed to get along on a memory which is equivalent to a printed page! When such a machine is properly run it will, in half an hour, do the work which a computing group of 20 people would do in about 2 or 3 years. Yet it's supposed to get along on a memory of one printed page. Imagine that you take 20 people, lock them up in a room for 3 years, provide them with 20 desk multipliers, and institute this rule: during the entire proceedings all of them together may never have more than one page written full. They can erase any amount, put it back again, but they are entitled at any given time only to one page. It's clear where the bottleneck of this process lies. The planning may be difficult, input and output may be cumbersome, and so on, but the main trouble is that it has a phenomenally low memory for the computing to be done. The whole technique of computing will be completely distorted by this modus operandi.

This is an extremely abnormal economy. By going to high speed, you cut yourself off from the efficient methods of storing information and push yourself into an inefficient one. A thousand-number computer memory is a very large object, an object which it took years to develop; all existing types are very much in an experimental stage at this moment, and none of them are small or cheap. Yet they are the equivalent of one printed page. The reason why one is forced to use these memories is this. [Each multiplication requires certain numbers from the memory and the product often goes to the memory. There are other arithmetic operations, and these require access to the memory. The orders to control these arithmetic operations come from the memory.] One probably needs anywhere between five and eight accesses to the memory for each multiplication. Thus it is unreasonable to get a millisecond multiplier unless you have a memory to which you can get access in something of the order of a tenth of a millisecond. Now to get access to a printed book takes seconds, to get access to anything punched or written on paper takes a fraction of a second. Since one needs an access time of something like one

ten-thousandth of a second, one is forced out of these efficient techniques of storing information, into a highly inefficient and expensive technique.

In comparing artificial with natural automata there is one very important thing we do not know: whether nature has ever been subject to this handicap, or whether natural organisms involve some much better memory device. That the secondary memory devices which humans have developed, namely, libraries, etc., are so vastly more efficient than this, is some reason to suspect that natural mechanisms for memory may also be quite as clumsy as the high speed memories with which we think we have to operate. But we know practically nothing about this.

Let me, before I close today, mention one more thing. In any fast machine the memory you need is characterized by two data: the capacity and the access time. [Von Neumann said that at that moment there was no technique for building a memory with both an adequate capacity and a sufficiently good access time. What is done is to construct a hierarchy of memories. The first memory has the required speed and is as large as you can make it, but it is not large enough. The second memory is much larger, but slower. Numbers are transferred from the second memory to the first memory when needed. There may be a third memory which is larger but slower, and so on. An electrostatic memory tube, a magnetic tape, and a card file would constitute such a hierarchy of memories. See *The Computer and the Brain* 33–37.]

Second Lecture

RIGOROUS THEORIES OF CONTROL AND INFORMATION

Theory of information: The strict part. The concept of information. The corresponding mathematical-logical concept of sets and partitions.

Close connection with formal logics. Alternative approach by way of model-automata. Common traits in these two approaches: All-or-none character. The work which connects these two approaches.

Methods of describing automata: Syntheses from components or treatment as a whole.

Synthetic approach: Nature of the element-organs. Their similarity to neurons. The program of McCulloch and Pitts: Formal neural networks. Their main result.

Treatment as a whole: Turing's theory of automata. The relationship of an automaton and of the mathematical problems that can be solved with its help. The concept of a universal automaton. Turing's main result.

Limitations of the McCulloch-Pitts and Turing automata. Input and output organs. Generalizations of these. Interpretation as sensor and motor organs.

[Von Neumann said that there are two parts to information theory: the rigorous and the probabilistic. The probabilistic part is probably more important for modern computing machinery, but the rigorous part is a necessary preliminary to it. The rigorous part of information theory is just a different way of dealing with formal logics.]

[He then explained some of the basic ideas of formal logics. He discussed briefly truth-functional connectives such as "and," "not," "if . . . then," and "not both," and their interdefinability. He explained the idea of a variable and the quantifiers "all" and "some." He concluded: "If you have this machinery you can express anything that is dealt with in mathematics; or that is dealt with in any subject, for that matter, as long as it's dealt with sharply."]

I am not going to go into this subject, because in order to make a theory of information, another machinery which is quite closely related to this but looks somewhat different, is more cogent. This is con-

nected with the work of McCulloch and Pitts,[1] on the one hand, and the logician Turing on the other.[2] Both endeavors in the subject replace formal logics as indicated here and as classically pursued, by the discussion of certain fictitious mechanisms or axiomatic paper automata, which are merely outlined, but which nobody is particularly concerned to build. Both of them show that their fictitious mechanisms are exactly co-extensional with formal logics; in other words, that what their automata can do can be described in logical terms and, conversely, that anything which can be described rigorously in logical terms can also be done by automata. [Von Neumann was assuming that a finite McCulloch-Pitts neuron net is supplied with an infinite blank tape. The result to which he referred is the equivalence of Turing computability, λ-definability, and general recursiveness. See Turing's "Computability and λ-Definability."]

I'm going to describe both the work of McCulloch and Pitts and the work of Turing, because they reflect two very important ways to get at the subject: the synthetic way, and the integral way. McCulloch and Pitts described structures which are built up from very simple elements, so that all you have to define axiomatically are the elements, and then their combination can be extremely complex. Turing started by axiomatically describing what the whole automaton is supposed to be, without telling what its elements are, just by describing how it's supposed to function.

The work of McCulloch and Pitts was definitely meant as a simple mathematical, logical model to be used in discussions of the human nervous system. That it wound up with something which is actually an equivalent of formal logics is very remarkable and was part of the point McCulloch and Pitts wanted to drive home, but only part of that point. Their model also has a meaning which concerns me at this moment a little less, but about which I will tell, without immediately stating where it ties in to formal logics. They wanted to discuss neurons. They took the position that they did not want to get tied up in the physiological and chemical complexities of what a neuron really is. They used what is known in mathematics as the axiomatic method, stating a few simple postulates and not being concerned with how nature manages to achieve such a gadget.

[1] [McCulloch and Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity." See also Secs. 1–7 of von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Burks and Wright, "Theory of Logical Nets," and Kleene, "Representation of Events in Nerve Nets and Finite Automata."]

[2] [Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem."]

They went one step further. This has been emphasized very strongly by those who criticize their work, although it seems to me that the extent to which they went further can be justified. They said that they did not want to axiomatize the neuron as it actually exists, but they wanted to axiomatize an idealized neuron, which is much simpler than the real one. They believed that the extremely amputated, simplified, idealized object which they axiomatized possessed the essential traits of the neuron, and that all else are incidental complications, which in a first analysis are better forgotten. Now, I am quite sure that it will be a long time before this point is generally agreed to by everybody, if ever; namely, whether or not what one overlooks in this simplification had really better be forgotten or not. But it's certainly true that one gets a quick understanding of a part of the subject by making this idealization.

The definition of what we call a neuron is this. One should perhaps call it a formal neuron, because it certainly is not the real thing, though it has a number of the essential traits of the real thing. A neuron will be symbolically designated by a circle, which symbolizes the body of the neuron, and a line branching out from the circle, which symbolizes the axon of the neuron. An arrow is used to indicate that the axon of one neuron is incident on the body of another. A neuron has two states: it's excited or not. As to what excitation is, one need not tell. Its main characteristic is its operational characteristic and that has a certain circularity about it: its main trait is that it can excite other neurons. Somewhere at the end of an involved network of neurons the excited neuron excites something which is not a neuron. For instance, it excites a muscle, which then produces physical motion; or it excites a gland which can produce a secretion, in which case you get a chemical change. So, the ultimate output of the excited state really produces phenomena which fall outside our present treatment. These phenomena will, for the sake of the present discussion, be entirely disregarded.

[Von Neumann stated the axioms governing the interaction of neurons. Following McCulloch and Pitts he assumed a uniform delay and for the time being disregarded "the important phenomenon of fatigue, the fact that after a neuron has been excited it is not usable for a while." Fatigue plays an important role in the functioning of an organism (see p. 48 below), but in spite of fatigue one can get continuous action by using a chain of neurons, each feeding its successor. Von Neumann defined the threshold of a neuron and introduced inhibitory synapses, symbolized by a circle (instead of an arrowhead).]

[Von Neumann next presented what he called "the important

result of McCulloch and Pitts." Imagine a black box with a number of inputs and a single output. Select two times, t_1 and t_2. Specify which patterns of inputs from time t_1 to time t_2 are to produce an output and which are not.] No matter how you formulate your conditions, you can always put a neural network in the box which will realize these conditions, which means that the generality of neural systems is exactly the same as the generality of logics. The fact that something has been done with the system means not more and not less than you know what you are talking about, that you can state it in a finite number of words unambiguously and rigorously. I will not attempt to give the proof, which like all proofs in formal logics is not quite easy to render. [We will sketch the proof very briefly. It follows from the construction that von Neumann referred to that every switching function (truth function, Boolean function) can be realized by a neural network with some fixed delay. A cyclic neural memory of arbitrary finite capacity can be attached to this switching net. When this composite network is augmented by an infinite tape, the result is a Turing machine. Moreover, corresponding to each Turing machine M, there is a network of this kind which computes the same number as M.]

[Von Neumann showed how to construct a few sample networks. The first is a network in which a dominates b, b dominates c, but c dominates a, shown in Figure 1. Each neuron is excited (fires) if it is stimulated on its excitatory input (the input with an arrow) but is not stimulated on its inhibitory input (the input with a small circle). Hence if a and b are both stimulated, output α will be active but not β; if b and c are both stimulated, output β will be active but not γ; while if a and c are both stimulated, output γ will be active but not α. Von Neumann used this network to illustrate a certain point. People have made statements about the non-quantitative character of human behavior, which statements seem to imply that in any quantitative mechanism, if a is stronger than b and b is stronger than c, then a is stronger than c. But in the above neural network a is stronger than b and b is stronger than c, while c is stronger than a.]

[Von Neumann then synthesized a number of other networks: simple memories, counters, and an elementary learning circuit. These are approximately the circuits of *Collected Works* 5.342–345. The learning circuit has two inputs, a and b. It counts the number of times a stimulus of a is followed by a stimulus of b. When this number reaches 256, the circuit emits a pulse whenever b is stimulated, independently of whether a is stimulated or not.] You see that you can produce circuits which look complicated, but which are actually quite

simple from the point of view of how they are synthesized and which have about the same complexity that they should have, namely, the complexity that grammar has. It is no more difficult to make this drawing up than to make up a sentence which describes what you want, and the essence of the result of McCulloch and Pitts is that there really isn't much difference between the two things. The rigorous verbal description is co-extensive with the description in terms of relay organs.

May I point out what follows from this from a philosophical point of view, and what does not follow. It certainly follows that anything that you can describe in words can also be done with the neuron method. And it follows that the nerves need not be supernaturally clever or complicated. In fact, they needn't be quite as clever and complicated as they are in reality, because an object which is a considerably amputated and emasculated neuron, which has many fewer attributes and responds in a much more schematic manner than a neuron, already can do everything you can think up.

What is not demonstrated by the McCulloch and Pitts result is equally important. It does not prove that any circuit you are designing in this manner really occurs in nature. It does not follow that the other functions of the nerve cell which have been dropped from this description are not essential. It does not follow that there is not a considerable problem left just in saying what you think is to be described. Let me try to put this in another way. If you consider certain activities of the human nervous system, you find that some of them are such that all parts of them can be described, but one is flabbergasted by the totality of what has to be described.

Suppose you want to describe the fact that when you look at a triangle you realize that it's a triangle, and you realize this whether it's small or large. It's relatively simple to describe geometrically what is meant: a triangle is a group of three lines arranged in a certain manner. Well, that's fine, except that you also recognize as a triangle something whose sides are curved, and a situation where only the vertices are indicated, and something where the interior is shaded and the exterior is not. You can recognize as a triangle many different things, all of which have some indication of a triangle in them, but the more details you try to put in a description of it the longer the description becomes.

In addition, the ability to recognize triangles is just an infinitesimal fraction of the analogies you can visually recognize in geometry, which in turn is an infinitesimal fraction of all the visual analogies you can recognize, each of which you can still describe. But with respect

to the whole visual machinery of interpreting a picture, of putting something into a picture, we get into domains which you certainly cannot describe in those terms. Everybody will put an interpretation into a Rorschach test, but what interpretation he puts into it is a function of his whole personality and his whole previous history, and this is supposed to be a very good method of making inferences as to what kind of a person he is.

In fine, now, all of this may seem a little arbitrary and accidental, but the basic fact involved is this, that our brains are exceedingly complicated. About one fifth of the brain is a visual brain, which, as far as we know, does nothing except make decisions about visual analogies. So, using the figures we have, which are not very good, but which are probably all right for an orientation, we conclude that apparently a network of about 2 billion relays does nothing but determine how to organize a visual picture. It is absolutely not clear a priori that there is any simpler description of what constitutes a visual analogy than a description of the visual brain.

Normally, a literary description of what an automaton is supposed to do is simpler than the complete diagram of the automaton. It is not true a priori that this will always be so. There is a good deal in formal logics to indicate that the description of the functions of an automaton is simpler than the automaton itself, as long as the automaton is not very complicated, but that when you get to high complications, the actual object is simpler than the literary description.

I am twisting a logical theorem a little, but it's a perfectly good logical theorem. It's a theorem of Gödel that the next logical step, the description of an object, is one class type higher than the object and is therefore asymptotically [?] infinitely longer to describe. I say that it's absolutely necessary; it's just a matter of complication when you get to this point. I think that there is a good deal of reason to suspect that this is so with things which have this disagreeably vague and fluid impression (like "What is a visual analogy?"), where one feels that one will never get to the end of the description. They may easily be in this condition already, where doing a thing is quicker than describing it, where the circuit is more quickly enumerated than a total description of all its functions in all conceivable conditions.

The insight that a formal neuron network can do anything which you can describe in words is a very important insight and simplifies matters enormously at low complication levels. It is by no means certain that it is a simplification on high complication levels. It is perfectly possible that on high complication levels the value of the theorem is in the reverse direction, that it simplifies matters because

it guarantees the reverse, namely, that you can express logics in terms of these efforts and the converse may not be true. [Von Neumann returned to this point on p. 51, after his discussion of Turing machines.]

[Von Neumann next discussed two cases in which circuits of idealized neurons do not seem to provide an explanation of how the nervous system actually performs a given function. The first case concerns the transmission by a nerve of a continuous number which represents some quantity such as blood pressure. The nerve does this by emitting pulses at a frequency which is a monotone function of the blood pressure. This behavior is explained in terms of neural fatigue: after a neuron responds, it is unable to respond for a certain period, called the refractory period, and the stronger the next stimulus the sooner it responds. He then raised the question "Why has the digital notation never been used in nature, as far as we know, and why has this pulse notation been used instead?" and said that this was the kind of question he was interested in. He suggested an answer: that the frequency modulation scheme is more reliable than the digital scheme. See Section 1.1.2.3 below, *The Computer and the Brain* 77–79, and *Collected Works* 5.306–308 and 5.375–376.]

[The second case in which circuits of idealized neurons do not seem to provide an explanation of how the nervous system actually performs a given function concerns memory. Von Neumann had earlier synthesized memory circuits from idealized neurons and he remarked that such memory circuits could be arbitrarily large. But he thought it probable that this is not the major mechanism used for memory in the nervous system.] This is not the way to make a memory for the simple reason that to use a switching organ like a neuron, or six to a dozen switching organs, as you actually would have to use because of fatigue, in order to do as small a thing as remember one binary digit, is a terrible waste, because a switching organ can do vastly more than store. In computing machines the classical example of a machine in which the switching organs were used to remember numbers is the ENIAC, an enormous gadget which has about 20 thousand vacuum tubes in it. The ENIAC is about five times larger than later machines which will presumably be far more efficient; it is an excellent machine in many ways, but it has one phenomenal shortcoming, namely, a very small memory. It has only a memory of 20 decimal numbers at points where it matters; in spite of this it is enormous. The reason is that vacuum tubes, in other words, switching organs, are used for that memory. All improvements on this machine postulate that some other components than standard vacuum tubes will be used for memory.

The memory requirements of the human nervous system are probably very large. Estimates have been made and they are of the order of 10^{16} binary digits. I will not attempt to justify this estimate; a great deal can be said for any count. I think there is a good deal of reason to believe that 10^{10} switching organs, which is about what we have, is probably not the right order of magnitude for storing the kind of memory that we use, and that it's probably best to admit that we simply do not know where the memory is. One can make all kinds of statements about it. One can surmise that the memory consists in a change of the synapses of the nerve cells, which is not decided by design. I don't know whether there is any good evidence for this, but I rather think there is not. You may suspect that the nerve cells contain a lot else than the switching trait, and that the memory sits there. It may be so, but I think that we simply know nothing. It may well be that the memory organs are of a completely different nature than the neurons.

The main difficulty with the memory organ is that it appears to be nowhere in particular. It is never very simple to locate anything in the brain, because the brain has an enormous ability to re-organize. Even when you have localized a function in a particular part of it, if you remove that part, you may discover that the brain has reorganized itself, reassigned its responsibilities, and the function is again being performed. The flexibility of the brain is very great, and this makes localization difficult. I suspect that the memory function is less localized than anything else. [Cf. *The Computer and the Brain* 63–68.]

I wanted to mention these two things [fatigue and memory] as very obvious lacunae in the McCulloch and Pitts approach to the nervous system. I want to talk next about the approach of Turing. In the McCulloch and Pitts theory the conclusion was that actual automata, properly described and axiomatized, are equivalent to formal logics. In Turing's theory the conclusion is the reverse. Turing was interested in formal logics, not in automata. He was concerned to prove certain theorems about an important problem of formal logics, the so-called *Entscheidungsproblem*, the problem of decision. The problem is to determine, for a class of logical expressions or propositions, whether there is a mechanical method for deciding whether an expression of this class is true or false. Turing's discussion of automata was really a formal, logical trick to deal with this problem in a somewhat more transparent and more consistent way than it had been dealt with before.

[Von Neumann then outlined Turing's definition of an automaton. Whereas McCulloch and Pitts started with components or elements, Turing started with states. At any time the automaton is in one of a

finite number of states. "The outside world" is a tape. The automaton senses one square of the tape, and it can change the contents of the square and move the tape one square to the left or right. A dictionary specifies, for each state and each tape symbol, what the next state will be and what will be done to the tape. The tape has a distinguished square. A finite program may be placed on the tape initially. The binary number computed by the automaton is recorded in alternate squares, starting with the distinguished square.]

[Von Neumann next described Turing's result concerning universal automata. There is a universal automaton $\bar{A}$ with the following properties: For each automaton A there is a sequence of instructions I_A such that for any sequence of instructions I, $\bar{A}$ supplied with both instructions I_A and I computes the same number as is computed by A supplied with instructions I.] $\bar{A}$ is able to imitate any automaton, even a much more complicated one. Thus a lesser degree of complexity in an automaton can be compensated for by an appropriate increase of complexity of the instructions. The importance of Turing's research is just this: that if you construct an automaton right, then any additional requirements about the automaton can be handled by sufficiently elaborate instructions. This is only true if A is sufficiently complicated, if it has reached a certain minimum level of complexity. In other words, a simpler thing will never perform certain operations, no matter what instructions you give it; but there is a very definite finite point where an automaton of this complexity can, when given suitable instructions, do anything that can be done by automata at all.

[Von Neumann then explained how the universal automaton $\bar{A}$ simulates an arbitrary automaton A. The instructions I_A contain a representation of the automaton A in the form of a dictionary, which tells, for each state of A and each tape symbol, the next state of A and what is to be done to the tape. The universal automaton $\bar{A}$ has the power to read any such dictionary and act on it. $\bar{A}$ writes on its tape, in sequence, the successive states of A and what is produced on the tape of A.] I will not go further in giving the details of this. I have gone into it to the point to which I did in order to point out that here, for the first time, one deals with something which has the attribute of universality, which has the ability to do anything that anybody can do. You also see that there is no vicious circle in it, because of the manner in which the extra complexity is brought in (by giving more elaborate instructions). You also see that the operation which ultimately leads to universality is connected with a rigorous theory of

how one describes objects and a rigorous routine of how to look up statements in a dictionary and obey them.

The formal logical investigations of Turing went a good deal further than this. Turing proved that there is something for which you cannot construct an automaton; namely, you cannot construct an automaton which can predict in how many steps another automaton which can solve a certain problem will actually solve it. So, you can construct an automaton which can do anything any automaton can do, but you cannot construct an automaton which will predict the behavior of any arbitrary automaton. In other words, you can build an organ which can do anything that can be done, but you cannot build an organ which tells you whether it can be done.

This is connected with the structure of formal logics and is specifically connected with a feature which I will not attempt to discuss, but which I would like to mention in the proper jargon for those of you who are familiar with modern formal logics. It is connected with the theory of types and with the results of Gödel. The feature is just this, that you can perform within the logical type that's involved everything that's feasible, but the question of whether something is feasible in a type belongs to a higher logical type. It's connected with the remark I made earlier (pp. 47–48): that it is characteristic of objects of low complexity that it is easier to talk about the object than produce it and easier to predict its properties than to build it. But in the complicated parts of formal logic it is always one order of magnitude harder to tell what an object can do than to produce the object. The domain of the validity of the question is of a higher type than the question itself.

[This is the end of von Neumann's Second Lecture. I will add a commentary on his last two paragraphs, beginning with some general remarks on Turing machines.

A Turing machine is basically a finite automaton with an indefinitely extendible tape. But there are many different ways of using a Turing machine. Let the squares of the tape be numbered 0, 1, 2, 3, $\cdots$, with even numbered squares reserved for working space and odd numbered squares reserved for the program or problem statement (if there is one) and for the answer. Let the answer symbols be zero and one, in addition to the blank; these could, of course, be coded sequences of two basic symbols, a blank and a mark. Assume finally that the machine writes the answer digits (zero and one) in successive answer squares.

A "concrete Turing machine" is a Turing machine which has a

finite "program" or problem statement on its tape initially. An "abstract Turing machine" is the class of all concrete Turing machines which contain a given finite automaton. We can think of an abstract Turing machine as a finite automaton with an indefinitely extendible blank tape on which any program or problem may be placed initially.

Concrete Turing machines may be divided into two classes: the circular and the circle-free. A "circular" machine prints a finite sequence of binary digits and "halts." A "circle-free" machine continues to print binary digits in alternate squares forever; we will speak of it as computing an infinite sequence.

Von Neumann discussed above a universal Turing machine which consists of a finite automaton $\bar{A}$ and an indefinitely extendible tape. A universal Turing machine is an abstract Turing machine which computes every sequence computed by Turing machines. More precisely, for each concrete Turing machine with finite automaton A and program I, there is a program I_A such that machine $\bar{A}$ with programs I_A and I computes the sequence computed by machine A with program I. A universal Turing machine can be characterized in another way. Let Γ be the class of finite and infinite sequences computed by concrete Turing machines. Then, every sequence of Γ is computed by the abstract Turing machine $\bar{A} + I_A + I$, where I_A and I vary over all programs. Since the concatenation of two programs is a program, every sequence of Γ is computed by the abstract Turing machine $\bar{A} + I$, where I varies over all programs.

A "decision machine" for a given class of questions is an abstract Turing machine which, when given a question of that class, prints a one if the answer to the question is "yes" and a zero if the answer to the question is "no."

The "halting problem" is the problem of deciding whether an arbitrary concrete Turing machine is circular (will halt sometime) or circle-free. Turing showed that the halting problem is undecidable, that is, that there is no decision machine for halting.[3] The proof is given below in Sec. 1.6.3.2 of Part II. Turing proved as a corollary to this that there is no decision machine for deciding whether an arbitrary concrete Turing machine will ever print a given symbol (for example, a zero). Since both halting and printing a given symbol are aspects of the behavior of a Turing machine, it follows from Turing's results that automata behavior is not completely predictable by automata. As von Neumann put it above, "you cannot construct an

[3] [Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," Sec. 8.]

automaton which will predict the behavior of any arbitrary automaton."

Concrete Turing machines can be enumerated and thereby placed in one to one correspondence with the non-negative integers. Consider all these machines and let the variable "t" range over the integers representing them. We define the number theoretic function $n(t)$ as the number of steps which machine t takes to print its first zero. If machine t never prints a zero, then $n(t)$ is defined to be zero.

Note that a sequence of n ones followed by a zero can be interpreted as the integer n. This leads to the question: Is there an abstract Turing machine which can compute $n(t)$ for any t? It follows immediately from Turing's corollary that there is not, for if we could compute $n(t)$ we could decide whether or not machine t ever prints a zero. I think that this is what von Neumann had in mind when he said "Turing proved that you cannot construct an automaton which can predict in how many steps another automaton which can solve a certain problem will actually solve it."

In the last paragraph of his Second Lecture von Neumann referred to a theorem of Gödel "that you can perform within the logical type that's involved everything that's feasible, but the question of whether something is feasible in a type belongs to a higher logical type." Since I knew of no such theorem by Gödel I found this reference puzzling, as well as the earlier reference to Gödel (p. 47) and a related reference in von Neumann's Hixon Symposium paper, "The General and Logical Theory of Automata" (*Collected Works* 5.310–311). I wrote Professor Kurt Gödel to see whether he could throw any light on it. His answer gives, I think, the most plausible explanation of the reference, and so I include the relevant parts of our correspondence, with minor editing.

I wrote to Professor Gödel as follows: "I am engaged in editing two of John von Neumann's uncompleted manuscripts on the theory of automata. In one of these, a series of lectures he delivered at the University of Illinois in 1949, he makes a reference to your work which I have been unable to figure out. Since there is the possibility he may have discussed this point with you, I am taking the liberty of writing you about it.

"The story begins with Johnny's Hixon Symposium talk at Pasadena in 1948. He discusses there the problem of giving a rigorous description of a visual analogy. In recognizing visual patterns, the human eye and nervous system function as a finite automaton with a certain behavior. Von Neumann seems to suggest that possibly the simplest way to describe the *behavior* of this finite automaton is to

describe the *structure* of the automaton itself. This is certainly plausible. But he then expresses the point in a way I do not understand: 'It is not at all certain that in this domain a real object might not constitute the simplest description of itself. That is, any attempt to describe it by the usual literary or formal-logical method may lead to something less manageable and more involved. In fact, *some results in modern logic* would tend to indicate that phenomena like this have to be expected when we come to really complicated entities.' The underlined passage seems to refer to your work. I enclose a copy of the full context.

"In his Illinois lectures, given in 1949, Johnny seems to be making the same point, namely, that the simplest way to describe accurately what *constitutes* a visual analogy is to specify the *connections* of the visual part of the brain. He then proceeds to say that there is a good deal in formal logic which indicates that when an automaton is not very complicated the description of the functions of that automaton is simpler than a description of the automaton itself but that the situation is reversed with respect to complicated automata. His reference to you then appears explicitly. He says, 'I am a little twisting a logical theorem, but it's a perfectly good logical theorem. It's a theorem of Gödel that the next logical step, the description of an object, is one class type higher than the object and is therefore asymptotically [?] infinitely longer to describe.'

"He returns to this point later after discussing Turing machines and mentioning Turing's result about the undecidability of the halting problem. He then says that all of this is connected with the theory of types and with your results. The recording transcript is mangled at this point and I will reconstruct it as best I can. 'It is connected with the theory of types and with the results of Gödel. The feature is just this, that you can perform within the logical type that's involved everything that's feasible but the question of whether something is feasible in a type belongs to a higher logical type. It's connected with the remark I made earlier: that it is characteristic of objects of low complexity that it is easier to talk about the object than produce it and easier to predict its properties than to build it. But in the complicated parts of formal logic it is always one order of magnitude harder to tell what an object can do than to produce the object. The domain of the validity of the question is of a higher type than the question itself.' I enclose copies of the relevant pages of the Illinois lectures.

"It is easy to regard the description of an object as of one type level higher than the object itself, but beyond this I do not see what von

Neumann has in mind. Two possibilities occured to me but both give a result opposite to that which Johnny needs. One may regard a Gödel number as a description of a formula. However, in some cases at least, the Gödel number of a formula may be described in fewer symbols than the formula, else the self-referring undecidable formula could not exist.[4] The other possibility concerns a theorem in your 1936 paper, "Über die Länge der Beweise." Given a system S and a larger system S_1. The theorem says that for every recursive function F, there exists a sentence which is provable in both systems and such that the shortest proofs in these two systems satisfy the inequality that the Gödel number of the proof in the smaller system is larger than the recursive function F applied to the Gödel number of the proof in the larger system. This fits everything that von Neumann says except that the result seems to go in the opposite direction: namely, the higher the type the shorter the proof.

"I would appreciate very much any light that you could throw on these puzzling passages of von Neumann."

Professor Gödel replied as follows. "I have some conjecture as to what von Neumann may have had in mind in the passages you quote, but since I never discussed these matters with him it is only a guess.

"I think the theorem of mine which von Neumann refers to is not that on the existence of undecidable propositions or that on the lengths of proofs but rather the fact that a complete epistemological description of a language A cannot be given in the same language A, because the concept of truth of sentences of A cannot be defined in A. It is this theorem which is the true reason for the existence of undecidable propositions in the formal systems containing arithmetic. I did not, however, formulate it explicitly in my paper of 1931 but only in my Princeton lectures of 1934.[5] The same theorem was proved by Tarski in his paper on the concept of truth published in 1933 in *Act. Soc. Sci. Lit. Vars.*, translated on pp. 152–278 of *Logic, Semantics, and Metamathematics.*[6]

"Now this theorem certainly shows that the description of what a mechanism is doing in certain cases is more involved than the descrip-

[4] [See Gödel's "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I." The undecidable formula has the Gödel number n and says "The formula whose Gödel number is n is not a theorem." Thus, via Gödel's coding, the undecidable formula refers to itself. It is undecidable in the sense that neither it nor its negation is a theorem of the system Gödel is studying.]

[5] [Gödel, "On Undecidable Propositions of Formal Mathematical Systems."]

[6] [The exact reference to Tarski's paper was added later.]

tion of the mechanism, in the sense that it requires new and more abstract primitive terms, namely higher types. However, this implies nothing as to the number of symbols necessary, where the relationship may very well be in the opposite direction, as you rightly remark.

"However, what von Neumann perhaps had in mind appears more clearly from the universal Turing machine. There it might be said that the complete description of its behavior is infinite because, in view of the non-existence of a decision procedure predicting its behavior, the complete description could be given only by an enumeration of all instances. Of course this presupposes that only decidable descriptions are considered to be complete descriptions, but this is in line with the finitistic way of thinking. The universal Turing machine, where the ratio of the two complexities is infinity, might then be considered to be a limiting case of other finite mechanisms. This immediately leads to von Neumann's conjecture."]

Third Lecture

STATISTICAL THEORIES OF INFORMATION

Theory of information: Probabilistic part. Relationship of strict and of probabilistic logics. Keynes' interpretation of probability theory. Exemplification of the relationship of logics to strict classical mechanics on the one hand, and to statistical mechanics on the other. Corresponding situation in quantum mechanics.

The mathematical aspects of the transition from strict to probabilistic logics. Analysis and combinatorics.

The thermodynamical aspect: Information and entropy.

The theory of Szilard.

The theory of Shannon.

Additional remarks on the thermodynamical nature of the internal balance of a computing machine.

I conclude my remarks about strict and rigorous questions of information at this point and pass on to statistical considerations involving information. That this is the important thing in dealing with automata and their functions is fairly evident, for two reasons at least. The first of these reasons may seem somewhat extraneous and accidental, although I think it is not, but the second reason is certainly not.

The first reason is that in no practical way can we imagine an automaton which is really reliable. If you axiomatize an automaton by telling exactly what it will do in every completely defined situation you are missing an important part of the problem. The axiomatization of automata for the completely defined situation is a very nice exercise for one who faces the problem for the first time, but everybody who has had experience with it knows that it's only a very preliminary stage of the problem.

The second reason for the importance of statistical considerations in the theory of automata is this. If you look at automata which have been built by men or which exist in nature you will very frequently notice that their structure is controlled only partly by rigorous requirements and is controlled to a much larger extent by the manner

in which they might fail and by the (more or less effective) precautionary measures which have been taken against their failure. And to say that they are precautions against failure is to overstate the case, to use an optimistic terminology which is completely alien to the subject. Rather than precautions against failure, they are arrangements by which it is attempted to achieve a state where at least a majority of all failures will not be lethal. There can be no question of eliminating failures or of completely paralyzing the effects of failures. All we can try to do is to arrange an automaton so that in the vast majority of failures it can continue to operate. These arrangements give palliatives of failures, not cures. Most of the arrangements of artificial and natural automata and the principles involved therein are of this sort.

To permit failure as an independent logical entity means that one does not state the axioms in a rigorous manner. The axioms are not of the form: if A and B happen, C will follow. The axioms are always of this variety: if A and B happen, C will follow with a certain specified probability, D will follow with another specified probability, and so on. In other words, in every situation several alternatives are permitted with various probabilities. Mathematically it is simplest to say that anything can follow upon anything in accordance with a probability matrix. You may put your question in this manner: If A and B have happened, what is the probability that C will follow? This probability pattern gives you a probabilistic system of logics. Both artificial and natural automata should be discussed in this system as soon as there is any degree of involvement.[1] I will come later to the question as to why it is just complexity which pushes one into this kind of axiomatization instead of a strict one.[2]

Now this inclines one to view probability as a branch of logics, or rather, to view logics affected with probability as an extension of ordinary rigorous logics. The view that probability is an extension of logics is not trivial, is not generally accepted, and is not the major interpretation of probability. It is, however, the classical interpretation. The competing interpretation is the frequency interpretation, the attitude that logic is completely rigorous, and with respect to phenomena about which you are not completely informed, you can only make statements of frequencies.

This distinction was, I think, quite clear to Laplace, who pointed

[1] [See von Neumann's "Probabilistic Logics and the Synthesis of Reliable Organs from Unreliable Components" for a detailed treatment of automata from this point of view.]

[2] [For a given probability of malfunction of a component, the more complex the automaton the more likely it is that a lethal failure will occur.]

out that there are two possible attitudes toward probability: the frequency and the logical.[3] In more recent times the distinction was emphasized strongly and made the basis of a system by the economist Keynes, who wrote his thesis on probability.[4] He analyzed this question in considerable detail and showed that, aside from the more conventional frequency viewpoint about probability, the logical one also exists. But he made no attempt to separate strict logics and probability and simply said that, if you view a sequence of events A and B, they have the quantitative characteristic, "the probability with which B follows A." The only tie to strict logics is that when the probability is one you have an implication, and when the probability is zero you have an exclusion, and when the probability is close to one or close to zero you can still make those inferences in a less rigorous domain.

There are undeniable weaknesses of the logical position. In some ways of looking at probability it is opportune not to identify zero probability with absurdity. Also, it is not quite clear in what sense a low probability means that one might expect that the thing will not happen. However, Keynes produced a self-consistent axiomatic system. There's a great deal in other modern theories, for instance, in quantum mechanics, which inclines one very strongly to take this philosophical position, although the last word about this subject has certainly not been said and is not going to be said for a long time. Anyway, one is also tempted in the case of quantum mechanics to modify one's outlook on logics and to view probability as intrinsically tied to logics.[5]

[Von Neumann discussed next two theories of probability and information "which are quite relevant in this context although they are not conceived from the strictly logical point of view." The first is the theory of entropy and information in thermodynamics; the second is Shannon's information theory.

In connection with entropy and information von Neumann referred to Boltzmann, Hartley, and Szilard. He explained at length the para-

[3] [*A Philosophical Essay on Probabilities.*]

[4] [*A Treatise on Probability.*]

[5] [In his "Quantum Logics (Strict-and-Probability-Logics)", von Neumann concluded: "*Probability logics cannot be reduced to strict logics, but constitute an essentially wider system than the latter, and statements of the form* $P(a, b) = \phi$ $(0 < \phi < 1)$ *are perfectly new and* sui generis *aspects of physical reality.*

"So probability logics appear as an essential extension of strict logics. This view, the so-called 'logical theory of probability' is the foundation of J. M. Keynes's work on this subject."

Compare von Neumann and Birkhoff, "The Logic of Quantum Mechanics," and von Neumann and Morgenstern, *Theory of Games and Economic Behavior*, Sec. 3.3.3.]

dox of Maxwell's demon and how Szilard resolved it by working out the relation of entropy to information.[6] Von Neumann said that Shannon's theory is a quantitative theory of measuring the capacity of a communications channel. He explained and illustrated the concept of redundancy. He pointed out that redundancy makes it possible to correct errors, for example, to read proof. Redundancy "is the only thing which makes it possible to write a text which is longer than, say, ten pages. In other words, a language which has maximum compression would actually be completely unsuited to conveying information beyond a certain degree of complexity, because you could never find out whether a text is right or wrong. And this is a question of principle. It follows, therefore, that the complexity of the medium in which you work has something to do with redundancy."

In his review of Wiener's *Cybernetics* von Neumann made an extended statement about entropy and information which it is appropriate to quote here. "Entropy for the physicist is a concept belonging to the discipline of thermodynamics where the transformations among the various forms of energy are studied. It is well known that the total energy of a complete, closed system is always conserved: energy is neither created nor lost but only transformed. This constitutes the first fundamental theorem of thermodynamics, or the energy theorem. There is, however, in addition, the second fundamental theorem of thermodynamics, or entropy theorem, which states that a hierarchy exists among the forms of energy: mechanical (kinetic or potential) energy, constituting the highest form, thermal energies constituting under it a decreasing hierarchical sequence in the order of decreasing temperature, and all other forms of energy permitting a complete classification relative to the gradations of this schema. It states, furthermore, that energy is always degraded, that is, that it always moves spontaneously from a higher form to a lower one, or if the opposite should happen in a part of the system, a compensatory degradation will have to take place in some other part. The bookkeeping that is required to account for this continuing overall degradation is effected by a certain well defined physical quantity, the entropy, which measures the hierarchic position held or the degree of degradation suffered by any form of energy.

"The thermodynamical methods of measuring entropy were known in the mid-nineteenth century. Already in the early work on statistical physics (L. Boltzmann, 1896) it was observed that entropy was closely

[6] [There is a good exposition of the work of Szilard, as well as that of Shannon and Hamming, in Brillouin's *Science and Information Theory*.]

connected with information: Boltzmann found entropy to be proportional to the logarithm of the number of alternatives which are possible for a physical system after all the information that one possesses about that system macroscopically (that is, on the directly, humanly observable scale) has been recorded.[7] In other words, it is proportional to the logarithm of the amount of missing information. This concept was elaborated further by various authors for various applications: H. Nyquist and R. V. L. Hartley, for transmission of information in the technical communication media (*Bell System Technical Journal*, Vol. 3, 1924, and Vol. 7, 1928); L. Szilard, for information in physics in general (*Zschr. f. Phys.*, Vol. 53, 1929); and the reviewer, for quantum mechanics and elementary particle physics (*Mathematical Foundations of Quantum Mechanics*, Berlin, 1932, Chapter V).

"The technically well-equipped reader is advised to consult at this point some additional literature, primarily L. Szilard's work, referred to above, which also contains a particularly instructive analysis of the famous thermodynamical paradox of "Maxwell's demon," and C. E. Shannon's very important and interesting recent work on the "Theory of Information," "Artificial Languages," "Codes," etc. (*Bell System Technical Journal*, Vol. 27, 1948). There is reason to believe that the general degeneration laws, which hold when entropy is used as a measure of the hierarchic position of energy, have valid analogs when entropy is used as a measure of information. On this basis one may suspect the existence of connections between thermodynamics and new extensions of logics."

In the Illinois lectures von Neumann next discussed Hamming's work on error-detecting and error-correcting codes. He then showed how the digital system with a base (binary, decimal, etc.) is an application of information theory. "Digitalization is just a very clever trick to produce extreme precision out of poor precision. By writing down 30 binary digits with 30 instruments, each of which is only good enough that you can distinguish two states of it (with intrinsic errors maybe on the 10 per cent level), you can represent a number to approximately one part in a billion. The main virtue of the digital system is that we know no other trick which can achieve this. From the information point of view it is clear that this can be done, because the entropy in 30 binary instruments is 30 units, and something which

[7] [*Vorlesungen über Gastheorie*, Vol. I, Sec. 6. Boltzmann's result appeared originally in 1877 in "Über die Beziehung zwischen dem zweiten Hauptsätze der mechanischen Wärmetheorie und der Wahrscheinlichkeitsrechnung respektive den Sätzen über das Wärmleichgewicht," *Wissenschaftliche Abhandlungen*, Vol. II, pp. 164–223.]

is known to one part in a billion has an entropy of the logarithm of a billion (to the base two), or about 30 units."

He then pointed out that while organisms use mixed analog-pulse systems for transmitting information, they never (to the best of our knowledge) use a coded digital system with a base. Rather, when "the nervous system transmits a number, it transmits it by what is essentially a frequency modulation trick and not as a coded digital aggregate." He suggested that the reason for this is that the frequency modulation method is more reliable than the digital system.]

I have been trying to justify the suspicion that a theory of information is needed and that very little of what is needed exists yet. Such small traces of it which do exist, and such information as one has about adjacent fields indicate that, if found, it is likely to be similar to two of our existing theories: formal logics and thermodynamics. It is not surprising that this new theory of information should be like formal logics, but it is surprising that it is likely to have a lot in common with thermodynamics.

Though this new theory of information will be similar to formal logics in many respects, it will probably be closer to ordinary mathematics than formal logics is. The reason for this is that present day formal logics has a very un-analytical, un-mathematical characteristic: it deals with absolutely all-or-none processes, where everything that either does or does not happen is finitely feasible or not finitely feasible. These all-or-none processes are only weakly connected to analysis, which is the best developed and best known part of mathematics, while they are closely connected to combinatorics, that part of mathematics of which we know the least. There is reason to believe that the kind of formal logical machinery we will have to use here will be closer to ordinary mathematics than present day logics is. Specifically, it will be closer to analysis, because all axioms are likely to be of a probabilistic and not of a rigorous character. Such a phenomenon has taken place in the foundations of quantum mechanics.

Thermodynamical concepts will probably enter into this new theory of information. There are strong indications that information is similar to entropy and that degenerative processes of entropy are paralleled by degenerative processes in the processing of information. It is likely that you cannot define the function of an automaton, or its efficiency, without characterizing the milieu in which it works by means of statistical traits like the ones used to characterize a milieu in thermodynamics. The statistical variables of the automaton's milieu will, of course, be somewhat more involved than the standard thermodynamical variable of temperature, but they will probably be similar in character.

Also, it is quite clear from the practice of building computing machines that the decisive properties of computing machines involve balance: balances between the speeds of various parts, balances between the speed of one part and the sizes of other parts, even balances between the speed ratio of two parts and the sizes of other parts. I mentioned this in the case of the hierarchic structure of memory [p. 41]. All of these requirements look like the balance requirements one makes in thermodynamics for the sake of efficiency. An automaton in which one part is too fast for another part, or where the memory is too small, or where the speed ratio of two memory stages is too large for the size of one, looks very much like a heat engine which doesn't run properly because excessively high temperature differences exist between its parts. I will not go into the details of this, but I would like to emphasize that this thermodynamical link is probably quite a close one.

THE ROLE OF HIGH AND OF EXTREMELY HIGH COMPLICATION

Comparisons between computing machines and the nervous systems. Estimates of size for computing machines, present and near future.

Estimates for size for the human central nervous system. Excursus about the "mixed" character of living organisms. Analog and digital elements. Observations about the "mixed" character of all componentry, artificial as well as natural. Interpretation of the position to be taken with respect to these.

Evaluation of the discrepancy in size between artificial and natural automata. Interpretation of this discrepancy in terms of physical factors. Nature of the materials used.

The probability of the presence of other intellectual factors. The role of complication and the theoretical penetration that it requires.

Questions of reliability and errors reconsidered. Probability of individual errors and length of procedure. Typical lengths of procedure for computing machines and for living organisms—that is, for artificial and for natural automata. Upper limits on acceptable probability of error in individual operations. Compensation by checking and self-correcting features.

Differences of principle in the way in which errors are dealt with in artificial and in natural automata. The "single error" principle in artificial automata. Crudeness of our approach in this case, due to the lack of adequate theory. More sophisticated treatment of this problem in natural automata: The role of the autonomy of parts. Connections between this autonomy and evolution.

After the broad general discussions of the last two lectures I would like to return to the subject of the specific automata which we know. I would like to compare artificial automata, specifically computing machines, with natural automata, particularly the human nervous system. In order to do this, I must say a few things in both cases about components and I must make certain comparisons of sizes.

As I mentioned before, in estimating the size of the human nervous system one is limited to a figure which is not very well established, but which is probably right in its order of magnitude. This is the

statement that there are 10^{10} neurons in the human brain. The number of nerves present elsewhere in the human organism is probably much smaller than this. Also, a large number of these other nerves originate in the brain anyway. The largest aggregation of nerves of the periphery is on the retina, and the optic nerve going from the retina to the brain is part of the brain.

Compared to this, the number of vacuum tubes involved in the computing machines we know of is very small, a million times smaller. The largest existing computing machine, the ENIAC, has 2×10^4 vacuum tubes. Another large computing machine, the SSEC, which belongs to the IBM Company, contains a mixture of vacuum tubes and relays, about 10 thousand of each. The fastest computing machines now under construction are designed to have several thousand vacuum tubes, perhaps 3 thousand. The reason for this difference in size between the ENIAC and the fast machines now under construction is a difference in the treatment of memory, which I will discuss later.

So the human nervous system is roughly a million times more complicated than these large computing machines. The increase in complexity from these computing machines to the central nervous system is more than the increase in complexity from a single vacuum tube to these computing machines. Even measuring complexity on a logarithmic scale, which is highly generous, we have not yet come half the way. I think that in any sensible definition of complexity, it would be much less than half way.

There is, however, a factor in favor of these machines: they're faster than the human brain. The time in which a human nerve can respond is about $\frac{1}{2}$ millisecond. However, that time is not a fair measure of the speed of the neuron, because what matters is not the time in which the neuron responds, but the time in which it recovers, the time from one response to the next potential response. That time is, at best, 5 milliseconds. In the case of a vacuum tube it's difficult to estimate the speed, but present designs call for repetition rates which are not much in excess of a million per second.

Thus the nervous system has a million times as many components as these machines have, but each component of the machine is about 5 thousand times faster than a neuron. Counting what can be done, hour by hour, the nervous system outperforms the machine by a factor of roughly 200. This estimate, however, favors the automaton, because an n-fold increase in size brings much more than an n-fold increase in what can be done. What can be done is a matter of the interrelationships between the components, and the number of

interrelationships increases with the square of the number of components. And apart from this, what can be done depends on certain minima. Below a certain minimum level of complexity you cannot do a certain thing, but above this minimum level of complexity you can do it.

[Von Neumann next compared the human nervous system and computers with respect to volume. The decisive factor is the space in which the control and amplifying functions are performed. In the case of the vacuum tube this is essentially the space between the cathode and the control grid, which is of the order of magnitude of a millimeter. In the case of the nerve cell it is the thickness of the nerve membrane, which is of the order of 1 micron. The ratio in size is about 1000 to 1, and this is also the ratio in voltage, so that the intensity of the field which is used for control and amplification is about the same in the vacuum tube and the nerve cell. This means that differences in total energy dissipation are mainly due to differences in size. "A discrepancy of 10^3 in linear size means a discrepancy of 10^9 in volume, and probably a not very different discrepancy in energy." See also *Collected Works* 5.299–302 and *The Computer and the Brain* 44–52.

He then calculated the energy which is dissipated "per elementary act of information, that is, per elementary decision of a two-way alternative and per elementary transmittal of 1 unit of information." He did this for three cases: the thermodynamical minimum, the vacuum tube, and the neuron.

In the third lecture he said that thermodynamical information is measured by the logarithm, to the base two, of the number of alternatives involved. The thermodynamical information in the case of two alternatives is thus one, "except that this is not the unit in which you measure energy. Entropy is energy only if you specify the temperature. So, running at low temperature you can say what energy should be dissipated." He then computed the thermodynamical minimum of energy per elementary act of information from the formula $kT \log_e N$ ergs, where k is Boltzmann's constant (1.4×10^{-16} ergs per degree), T is the temperature in absolute units, and N is the number of alternatives. For a binary act $N = 2$, and taking the temperature to be about 300 degrees absolute, he obtained 3×10^{-14} ergs for the thermodynamical minimum.

Von Neumann then estimated that the brain dissipates 25 watts, has 10^{10} neurons, and that on the average a neuron is activated about 10 times per second. Hence the energy dissipation per binary act in a nerve cell is roughly 3×10^{-3} ergs. He estimated that a vacuum tube dissipates 6 watts, is activated about 100,000 times per second, and thus dissipates 6×10^2 ergs per binary act.]

So our present machinery is about 200 thousand times less efficient than the nervous system is. Computing machines will be improved in the next few years, perhaps by replacing vacuum tubes with amplifying crystals, but even then they will be of the order of 10 thousand times less efficient than the nervous system. The remarkable thing, however, is the enormous gap between the thermodynamical minimum (3×10^{-14} ergs) and the energy dissipation per binary act in the neuron (3×10^{-3} ergs). The factor here is 10^{11}. This shows that the thermodynamical analysis is missing a large part of the story. Measured on a logarithmic scale, the gap between our instrumentation, which is obviously amateurish, and the procedures of nature, which show a professional touch, is about half the gap between the best devices we know about and the thermodynamical minimum. What this gap is due to I don't know. I suspect that it's due to something like a desire for reliability of operation.

Thus, for an elementary act of information, nature does not use what, from the point of view of physics, is an elementary system with two stable states, such as a hydrogen atom. All the switching organs used are much larger. If nature really operated with these elementary systems, switching organs would have dimensions of the order of a few angstroms, while the smallest switching organs we know have dimensions of the order of thousands or tens of thousands of angstroms. There is obviously something which forces one to use organs several orders of magnitude larger than is required by the strict thermodynamical argument. Thus, though the observation that information is entropy tells an important part of the story, it by no means tells the whole story. There is a factor of 10^{11} still to be accounted for.

[Von Neumann then discussed memory components. Vacuum tubes, which are switching organs, may be used for memory. But since the standard circuit for storing a binary digit has two tubes, and additional tubes are needed for transmitting the information in and out, it is not feasible to build a large memory out of vacuum tubes. "The actual devices which are used are of such a nature that the store is effected, not in a macroscopic object like a vacuum tube, but in something which is microscopic and has only a virtual existence." Von Neumann describes two devices of this sort: acoustic delay line storage and cathode ray tube storage.

An acoustic delay line is a tube which is filled with a medium such as mercury and which has a piezo-electric crystal at each end. When the transmitting crystal is stimulated electrically, it produces an acoustic wave that travels through the mercury and causes the receiving crystal to produce an electrical signal. This signal is amplified, reshaped, and retimed and sent to the transmitting crystal again.

This acoustic-electrical cycle can be repeated indefinitely, thereby providing storage. A binary digit is represented by the presence or absence of a pulse at a given position at a given time, and since the pulses circulate around the system, the digit is not stored in any fixed position. "The thing which remembers is nowhere in particular."

Information may be stored in a cathode ray tube in the form of electric charges on the inside surface of the tube. A binary digit is represented by the charge stored in a small area. These charges are deposited and sensed by means of the electron beam of the cathode ray tube. Since the area associated with a given binary digit must be recharged frequently, and since this area may be moved by changing the position of the electron beam, this memory is also virtual. "The site of the memory is really nowhere organically, and the mode of control produces the memory organ in a virtual sense, because no permanent physical changes ever occur."]

There's therefore no reason to believe that the memory of the central nervous system is in the switching organs (the neurons). The size of the human memory must be very great, much greater than 10^{10} binary units. If you count the impressions which a human gets in his life or other things which appear to be critical, you obtain numbers like 10^{15}. One cannot place much faith in these estimates, but I think it likely that the memory capacity of the human nervous system is greater than 10^{10}. I don't know how legitimate it is to transfer our experience with computing machines to natural systems, but if our experience is worth anything it is highly unlikely that the natural memory should be in switching organs or should consist of anything as unsophisticated and crude as the modification of a switching organ. It has been suggested that memory consists in a change of threshold at a synapse. I don't know if this is true, but the memory of computing machines does not consist of bending a grid. A comparison between artificial automata and the central nervous system makes it probable that the memory of the latter is more sophisticated and more virtual than this. Therefore, I think that all guesses about what the memory of the human organism is, and where it sits, are premature.

Another thing of which I would like to talk is this. I have been talking as if a nerve cell were really a pure switching organ. It has been pointed out by many experts in neurology and adjacent fields that the nerve cell is not a pure switching organ but a very delicate continuous organ. In the lingo of computing machinery one would say it is an analog device that can do vastly more than transmit or not transmit a pulse. There is a possible answer to this, namely, that vacuum tubes, electromechanical relays, etc. are not switching devices

either, since they have continuous properties. They are all characterized by this, however, that there is at least one way to run them where they have essentially an all-or-none response. What matters is how the component runs when the organism is functioning normally. Now nerve cells do not usually run as all-or-none organs. For instance, the method of translating a stimulus intensity into a frequency of response depends on fatigue and the time of recovery, which is a continuous or analog response. However, it is quite clear that the all-or-none character of a neuron is a very important part of the story.

The human organism is not a digital organ either, though one part of it, the nervous system, is essentially digital. Almost all the nervous stimuli end in organs which are not digital, such as a contracting muscle or an organ which causes secretions to produce a chemical. To control the production of a chemical and rely on the diffusion rate of a chemical is to employ a much more sophisticated analog procedure than we ever use in analog computing machines. The most important loops in the human system are of this nature. A system of nervous stimuli goes through a complicated network of nerves and then controls the operation of what is essentially a chemical factory. The chemicals are distributed by a very complicated hydrodynamical system, which is completely analog. These chemicals produce nervous stimuli which travel in a digital manner through the nervous system. There are loops where this change from digital into analog occurs several times. So the human organism is essentially a mixed system. But this does not decrease the necessity for understanding the digital part of it.

Computing machines aren't purely digital either. The way we run them now, their inputs and outputs are digital. But it's quite clear that we need certain non-digital inputs and outputs. It's frequently desirable to display the result, not in digits, but, say, as a curve on an oscilloscope screen. This is an analog output. Moreover, I think that the important applications of these devices will come when you can use them to control complicated machinery, for example, the flight of a missile or of a plane. In this case the inputs will come from an analog source and the outputs will control an analog process. This whole trans-continuous alternation between digital and analog mechanisms is probably characteristic of every field.

The digital aspect of automata should be emphasized at the present time, for we now have some logical tools to deal with digital mechanisms, and our understanding of digital mechanisms is behind our understanding of analog mechanisms. Also, it appears that digital

mechanisms are necessary for complicated functions. Pure analog mechanisms are usually not suited for very complicated situations. The only way to handle a complicated situation with analog mechanisms is to break it up into parts and deal with the parts separately and alternately, and this is a digital trick.

Let me now come to the following question. Our artificial automata are much smaller than natural automata in what they do and in the number of components they have, and they're phenomenally more expensive in terms of space and energy. Why is this so? It's manifestly hopeless to produce a true answer at the present time: We can hardly explain why two objects are different if we understand one a little and the other not at all. However, there are some obvious discrepancies in the tools with which we operate, which make it clear that we would have difficulty in going much further with these tools.

The materials which we are using are by their very nature not well suited for the small dimensions nature uses. Our combinations of metals, insulators, and vacuums are much more unstable than the materials used by nature; that they have higher tensile strengths is completely incidental. If a membrane is damaged it will reconstruct itself, but if a vacuum tube develops a short between its grid and cathode it will not reconstruct itself. Thus the natural materials have some sort of mechanical stability and are well balanced with respect to mechanical properties, electrical properties, and reliability requirements. Our artificial systems are patchworks in which we achieve desirable electrical traits at the price of mechanically unsound things. We use techniques which are excellent for fitting metal to metal but are not very good for fitting metal to vacuum. To obtain millimeter spacings in an inaccessible vacuum space is a great mechanical achievement, and we will not be able to decrease the size by large factors here. And so the differences in size between artificial and natural automata are probably connected essentially with quite radical differences in materials.

[Von Neumann proceeded to discuss what he thought was a deeper cause of the discrepancy in size between natural and artificial automata. This is that many of the components of the natural system serve to make the system reliable. As he noted in the third lecture, actual computing elements function correctly with a certain probability only, not with certainty. In small systems the probability that the whole system will behave incorrectly is relatively small and may often be neglected, but this is not the case with large systems. Thus error considerations become more important as the system becomes more complex.

Von Neumann made some very rough calculations to justify this

conclusion. Assuming that the system is designed in such a way that the failure of a single element would result in failure of the whole system, he calculated the error probability required for a given mean free path between system errors. For the human nervous system he used the following figures: 10^{10} neurons; each neuron activated 10 times per second on the average; a mean free path between fatal errors of 60 years (the average life span). Since 60 years is about 2×10^9 seconds, the product of these numbers is 2×10^{20}. Hence an error probability of 0.5×10^{-20} for each activation of an element is required under these assumptions. For a digital computer he used the figures: 5×10^3 vacuum tubes, 10^5 activations per tube per second, and a desired mean free path between system errors of 7 hours (about 2×10^4 seconds). An error probability of 10^{-13} per tube activation is required for this degree of reliability. Compare the calculations at *Collected Works* 5.366–367.

He pointed out that vacuum tubes, and artificial components generally, do not have an error probability as low as 10^{-13}, and that neurons probably do not either. We try to design computing machines so that they will stop when they make an error and the operator can then locate it and correct it. For example, a computer may perform a certain operation twice, compare the results, and stop if the results differ.]

It's very likely that on the basis of the philosophy that every error has to be caught, explained, and corrected, a system of the complexity of the living organism would not run for a millisecond. Such a system is so well integrated that it can operate across errors. An error in it does not in general indicate a degenerative tendency. The system is sufficiently flexible and well organized that as soon as an error shows up in any part of it, the system automatically senses whether this error matters or not. If it doesn't matter, the system continues to operate without paying any attention to it. If the error seems to the system to be important, the system blocks that region out, by-passes it, and proceeds along other channels. The system then analyzes the region separately at leisure and corrects what goes on there, and if correction is impossible the system just blocks the region off and by-passes it forever. The duration of operability of the automaton is determined by the time it takes until so many incurable errors have occurred, so many alterations and permanent by-passes have been made, that finally the operability is really impaired. This is a completely different philosophy from the philosophy which proclaims that the end of the world is at hand as soon as the first error has occurred.

To apply the philosophy underlying natural automata to artificial

automata we must understand complicated mechanisms better than we do, we must have more elaborate statistics about what goes wrong, and we must have much more perfect statistical information about the milieu in which a mechanism lives than we now have. An automaton can not be separated from the milieu to which it responds. By that I mean that it's meaningless to say that an automaton is good or bad, fast or slow, reliable or unreliable, without telling in what milieu it operates. The characteristics of a human for survival are well defined on the surface of the earth in its present state, though for most types of humans you must actually specialize the situation a little further than this. But it is meaningless to argue how the human would survive on the bottom of the ocean or in a temperature of 1000 degrees centigrade. Similarly, in discussing a computing machine it is meaningless to ask how fast or how slow it is, unless you specify what type of problems will be given to it.

It makes an enormous difference whether a computing machine is designed, say, for more or less typical problems of mathematical analysis, or for number theory, or combinatorics, or for translating a text. We have an approximate idea of how to design a machine to handle the typical general problems of mathematical analysis. I doubt that we will produce a machine which is very good for number theory except on the basis of our present knowledge of the statistical properties of number theory. I think we have very little idea as to how to design good machines for combinatorics and translation.

What matters is that the statistical properties of problems of mathematical analysis are reasonably well known, and as far as we know, reasonably homogeneous. Consider some problems in mathematical analysis which look fairly different from each other and which by mathematical standards are very different: finding the roots of an equation of the tenth order, inverting a matrix of the twentieth order, solving a proper value problem, solving an integral equation, or solving an integral differential equation. These problems are surprisingly homogeneous with respect to the statistical properties which matter for a computing machine: the fraction of multiplications to other operations, the number of memory references per multiplication, and the optimal hierarchic structure of the memory with respect to access time. There's vastly less homogeneity in number theory. There are viewpoints under which number theory is homogeneous, but we don't know them.

So, it is true for all these automata that you can only assign them a value in combination with the milieu which they have to face. Natural automata are much better suited to their milieu than any

artifacts we know. It is therefore quite possible that we are not too far from the limits of complication which can be achieved in artificial automata without really fundamental insights into a theory of information, although one should be very careful with such statements because they can sound awfully ridiculous 5 years later.

[Von Neumann then explained why computing machines are designed to stop when a single error occurs. The fault must be located and corrected by the engineer, and it is very difficult for him to localize a fault if there are several of them. If there is only one fault he can often divide the machine into two parts and determine which part made the error. This process can be repeated until he isolates the fault. This general method becomes much more complicated if there are two or three faults, and breaks down when there are many faults.]

The fact that natural organisms have such a radically different attitude about errors and behave so differently when an error occurs is probably connected with some other traits of natural organisms, which are entirely absent from our automata. The ability of a natural organism to survive in spite of a high incidence of error (which our artificial automata are incapable of) probably requires a very high flexibility and ability of the automaton to watch itself and reorganize itself. And this probably requires a very considerable autonomy of parts. There is a high autonomy of parts in the human nervous system. This autonomy of parts of a system has an effect which is observable in the human nervous system but not in artificial automata. When parts are autonomous and able to reorganize themselves, when there are several organs each capable of taking control in an emergency, an antagonistic relation can develop between the parts so that they are no longer friendly and cooperative. It is quite likely that all these phenomena are connected.

Fifth Lecture

RE-EVALUATION OF THE PROBLEMS OF COMPLICATED AUTOMATA— PROBLEMS OF HIERARCHY AND EVOLUTION

Analysis of componentry and analysis of integration. Although these parts have to appear together in a complete theory, the present state of our information does not justify this yet.

The first problem: Reasons for not going into it in detail here. Questions of principle regarding the nature of relay organs.

The second problem: Coincides with a theory of information and of automata. Reconsideration of the broader program regarding a theoretical discussion of automata as indicated at the end of the second lecture.

Synthesis of automata. Automata which can effect such syntheses.

The intuitive concept of "complication." Surmise of its degenerative character: In connection with descriptions of processes by automata and in connection with syntheses of automata by automata.

Qualifications and difficulties regarding this concept of degeneracy.

Rigorous discussion: Automata and their "elementary" parts. Definition and listing of elementary parts. Synthesis of automata by automata. The problem of self-reproduction.

Main types of constructive automata which are relevant in this connection: The concept of a general instruction. The general constructive automaton which can follow an instruction. The general copying automaton. The self-reproducing combination.

Self-reproduction combined with synthesis of other automata: The enzymatic function. Comparison with the known major traits of genetic and mutation mechanisms.

The questions on which I've talked so far all bear on automata whose operations are not directed at themselves, so that they produce results which are of a completely different character than themselves. This is obvious in each of the three cases I have referred to.

It is evident in the case of a Turing automaton, which is a box with a finite number of states. Its outputs are modifications of another entity, which, for the sake of convenience, I call a punched tape.

This tape is not itself an object which has states between which it can move of its own accord. Furthermore, it is not finite, but is assumed to be infinite in both directions. Thus this tape is qualitatively completely different from the automaton which does the punching, and so the automaton is working into a qualitatively different medium.

This is equally true for the automata discussed by McCulloch and Pitts, which are made of units, called neurons, that produce pulses. The inputs and outputs of these automata are not the neurons but the pulses. It is true that these pulses may go to peripheral organs, thereby producing entirely different reactions. But even there one primarily thinks, say, of feeding the pulses into motor or secretory organs, so it is still true that the inputs and outputs are completely different from the automaton itself.

Finally, it is entirely true for computing machines, which can be thought of as machines which are fed, and emit, some medium like punched tape. Of course, I do not consider it essentially different whether the medium is a punched card, a magnetic wire, a magnetized metal tape with many channels on it, or a piece of film with points photographed on it. In all these cases the medium which is fed to the automaton and which is produced by the automaton is completely different from the automaton. In fact, the automaton doesn't produce any medium at all; it merely modifies a medium which is completely different from it. One can also imagine a computing machine with an output of pulses which are fed to control completely different entities. But again, the automaton is completely different from the electrical pulses it emits. So there's this qualitative difference.

A complete discussion of automata can be obtained only by taking a broader view of these things and considering automata which can have outputs something like themselves. Now, one has to be careful what one means by this. There is no question of producing matter out of nothing. Rather, one imagines automata which can modify objects similar to themselves, or effect syntheses by picking up parts and putting them together, or take synthesized entities apart. In order to discuss these things, one has to imagine a formal set-up like this. Draw up a list of unambiguously defined elementary parts. Imagine that there is a practically unlimited supply of these parts floating around in a large container. One can then imagine an automaton functioning in the following manner: It also is floating around in this medium; its essential activity is to pick up parts and put them together, or, if aggregates of parts are found, to take them apart.

This is an axiomatically shortened and simplified description of

what an organism does. It's true that this view has certain limitations, but they are not fundamentally different from the inherent limitations of the axiomatic method. Any result one might reach in this manner will depend quite essentially on how one has chosen to define the elementary parts. It is a commonplace of all axiomatic methods that it is very difficult to give rigorous rules as to how one should choose the elementary parts, so that whether the choice of the elements was reasonable is a matter of common sense judgment. There is no rigorous description of what choice is reasonable and what choice is not.

First of all, one may define parts in such numbers, and each of them so large and involved, that one has defined the whole problem away. If you chose to define as elementary objects things which are analogous to whole living organisms, then you obviously have killed the problem, because you would have to attribute to these parts just those functions of the living organism which you would like to describe or to understand. So, by choosing the parts too large, by attributing too many and too complex functions to them, you lose the problem at the moment of defining it.

One also loses the problem by defining the parts too small, for instance, by insisting that nothing larger than a single molecule, single atom, or single elementary particle will rate as a part. In this case one would probably get completely bogged down in questions which, while very important and interesting, are entirely anterior to our problem. We are interested here in organizational questions about complicated organisms, and not in questions about the structure of matter or the quantum mechanical background of valency chemistry. So, it is clear that one has to use some common sense criteria about choosing the parts neither too large nor too small.

Even if one chooses the parts in the right order of magnitude, there are many ways of choosing them, none of which is intrinsically much better than any other. There is in formal logics a very similar difficulty, that the whole system requires an agreement on axioms, and that there are no rigorous rules on how axioms should be chosen, just the common sense rules that one would like to get the system one is interested in and would not like to state in his axioms either things which are really terminal theorems of his theory or things which belong to vastly anterior fields. For example, in axiomatizing geometry one should assume theorems from set theory, because one is not interested in how to get from sets to numbers, or from numbers to geometry. Again, one does not choose the more sophisticated theorems of analytic number theory as axioms of geometry, because one wants to cut in at an earlier point.

Even if the axioms are chosen within the common sense area, it is usually very difficult to achieve an agreement between two people who have done this independently. For instance, in the literature of formal logics there are about as many notations as there are authors, and anybody who has used a notation for a few weeks feels that it's more or less superior to any other. So, while the choice of notations, of the elements, is enormously important and absolutely basic for an application of the axiomatic method, this choice is neither rigorously justifiable nor humanly unambiguously justifiable. All one can do is to try to submit a system which will stand up under common sense criteria. I will give an indication of how one system can be constructed, but I want to emphasize very strongly how relatively I state this system.

I will introduce as elementary units neurons, a "muscle," entities which make and cut fixed contacts, and entities which supply energy, all defined with about that degree of superficiality with which the formal theory of McCulloch and Pitts describes an actual neuron. If you describe muscles, connective tissues, "disconnecting tissues," and means of providing metabolic energy, all with this degree of schematization, you wind up with a system of elements with which you can work in a reasonably uncomplicated manner. You probably wind up with something like 10 or 12 or 15 elementary parts.

By axiomatizing automata in this manner, one has thrown half of the problem out the window, and it may be the more important half. One has resigned oneself not to explain how these parts are made up of real things, specifically, how these parts are made up of actual elementary particles, or even of higher chemical molecules. One does not ask the most intriguing, exciting, and important question of why the molecules or aggregates which in nature really occur in these parts are the sort of things they are, why they are essentially very large molecules in some cases but large aggregations in other cases, why they always lie in a range beginning at a few microns and ending at a few decimeters. This is a very peculiar range for an elementary object, since it is, even on a linear scale, at least five powers of ten away from the sizes of really elementary entities.

These things will not be explained; we will simply assume that elementary parts with certain properties exist. The question that one can then hope to answer, or at least investigate, is: What principles are involved in organizing these elementary parts into functioning organisms, what are the traits of such organisms, and what are the essential quantitative characteristics of such organisms? I will discuss the matter entirely from this limited point of view.

[At this point von Neumann made the remarks on information, logic, thermodynamics, and balance which now appear at the end of the Third Lecture. They are placed there because that is where von Neumann's detailed outline located them. Those remarks are relevant to the present discussion because the concept of complication which von Neumann introduced next belongs to information theory.]

There is a concept which will be quite useful here, of which we have a certain intuitive idea, but which is vague, unscientific, and imperfect. This concept clearly belongs to the subject of information, and quasi-thermodynamical considerations are relevant to it. I know no adequate name for it, but it is best described by calling it "complication." It is effectivity in complication, or the potentiality to do things. I am not thinking about how involved the object is, but how involved its purposive operations are. In this sense, an object is of the highest degree of complexity if it can do very difficult and involved things.

I mention this because when you consider automata whose normal function is to synthesize other automata from elementary parts (living organisms and such familiar artificial automata as machine tools), you find the following remarkable thing. There are two states of mind, in each of which one can put himself in a minute, and in each of which we feel that a certain statement is obvious. But each of these two statements is the opposite or negation of the other!

Anybody who looks at living organisms knows perfectly well that they can produce other organisms like themselves. This is their normal function, they wouldn't exist if they didn't do this, and it's plausible that this is the reason why they abound in the world. In other words, living organisms are very complicated aggregations of elementary parts, and by any reasonable theory of probability or thermodynamics highly improbable. That they should occur in the world at all is a miracle of the first magnitude; the only thing which removes, or mitigates, this miracle is that they reproduce themselves. Therefore, if by any peculiar accident there should ever be one of them, from there on the rules of probability do not apply, and there will be many of them, at least if the milieu is reasonable. But a reasonable milieu is already a thermodynamically much less improbable thing. So, the operations of probability somehow leave a loophole at this point, and it is by the process of self-reproduction that they are pierced.

Furthermore, it's equally evident that what goes on is actually one degree better than self-reproduction, for organisms appear to have gotten more elaborate in the course of time. Today's organisms are phylogenetically descended from others which were vastly simpler

than they are, so much simpler, in fact, that it's inconceivable how any kind of description of the later, complex organism could have existed in the earlier one. It's not easy to imagine in what sense a gene, which is probably a low order affair, can contain a description of the human being which will come from it. But in this case you can say that since the gene has its effect only within another human organism, it probably need not contain a complete description of what is to happen, but only a few cues for a few alternatives. However, this is not so in phylogenetic evolution. That starts from simple entities, surrounded by an unliving amorphous milieu, and produces something more complicated. Evidently, these organisms have the ability to produce something more complicated than themselves.

The other line of argument, which leads to the opposite conclusion, arises from looking at artificial automata. Everyone knows that a machine tool is more complicated than the elements which can be made with it, and that, generally speaking, an automaton A, which can make an automaton B, must contain a complete description of B and also rules on how to behave while effecting the synthesis. So, one gets a very strong impression that complication, or productive potentiality in an organization, is degenerative, that an organization which synthesizes something is necessarily more complicated, of a higher order, than the organization it synthesizes. This conclusion, arrived at by considering artificial automata, is clearly opposite to our early conclusion, arrived at by considering living organisms.

I think that some relatively simple combinatorial discussions of artificial automata can contribute to mitigating this dilemma. Appealing to the organic, living world does not help us greatly, because we do not understand enough about how natural organisms function. We will stick to automata which we know completely because we made them, either actual artificial automata or paper automata described completely by some finite set of logical axioms. It is possible in this domain to describe automata which can reproduce themselves. So at least one can show that on the site where one would expect complication to be degenerative it is not necessarily degenerative at all, and, in fact, the production of a more complicated object from a less complicated object is possible.

The conclusion one should draw from this is that complication is degenerative below a certain minimum level. This conclusion is quite in harmony with other results in formal logics, to which I have referred a few times earlier during these lectures.[1] We do not now know

[1] [See the end of the Second Lecture.]

what complication is, or how to measure it, but I think that something like this conclusion is true even if one measures complication by the crudest possible standard, the number of elementary parts. There is a minimum number of parts below which complication is degenerative, in the sense that if one automaton makes another the second is less complex than the first, but above which it is possible for an automaton to construct other automata of equal or higher complexity. Where this number lies depends upon how you define the parts. I think that with reasonable definitions of parts, like those I will partially indicate later, which give one or two dozen parts with simple properties, this minimum number is large, in the millions. I don't have a good estimate of it, although I think that one will be produced before terribly long, but to do so will be laborious.

There is thus this completely decisive property of complexity, that there exists a critical size below which the process of synthesis is degenerative, but above which the phenomenon of synthesis, if properly arranged, can become explosive, in other words, where syntheses of automata can proceed in such a manner that each automaton will produce other automata which are more complex and of higher potentialities than itself.

Now, none of this can get out of the realm of vague statement until one has defined the concept of complication correctly. And one cannot define the concept of complication correctly until one has seen in greater detail some critical examples, that is, some of the constructs which exhibit the critical and paradoxical properties of complication. There is nothing new about this. It was exactly the same with conservation and non-conservation properties in physics, with the concepts of energy and entropy, and with other critical concepts. The simplest mechanical and thermodynamic systems had to be discussed for a long time before the correct concepts of energy and entropy could be abstracted from them.

[Von Neumann only briefly described the kinds of elements or parts he planned to use. There are neurons like those of McCulloch and Pitts. There are elements "that have absolutely no function except that they are rigid and produce a geometrical tie between their ends." Another kind of element is called a "motor organ" and a "muscle-like affair"; it contracts to zero length when stimulated. There is an organ which, when pulsed, "can either make or break a connection." He said that less than a dozen kinds of elements are needed. An automaton composed of these parts can catch other parts which accidentally come in contact with it; "it is possible to invent a system by which it can sense" what part it has caught.

In June of 1948 von Neumann gave three lectures on automata at the Institute for Advanced Study to a small group of friends. He probably did this in preparation for the Hixon Symposium which took place in September of that year.[2] These lectures contained the most detailed description of the parts of his self-reproducing automaton that I know of. For this reason, I have attempted to reconstruct, from the notes and memories of the audience, what he said about these parts and how they would function.

Von Neumann described eight kinds of parts. All seem to have been symbolized with straight lines; inputs and outputs were indicated at the ends and/or the middle. The temporal reference frame was discrete, each element taking a unit of time to respond. It is not clear whether he intended this list to be complete; I suspect that he had not yet made up his mind on this point.

Four of the parts perform logical and information processing operations. A *stimulus organ* receives and transmits stimuli; it receives them disjunctively, that is, it realizes the truth-function "p or q." A *coincidence organ* realizes the truth-function "p and q." An *inhibitory organ* realizes the truth-function "p and not-q." A *stimuli producer* serves as a source of stimuli.

The fifth part is a *rigid member*, from which a rigid frame for an automaton can be constructed. A rigid member does not carry any stimuli; that is, it is an insulated girder. A rigid member may be connected to other rigid members as well as to parts which are not rigid members. These connections are made by a *fusing organ* which, when stimulated, welds or solders two parts together. Presumably the fusing organ is used in the following way. Suppose point a of one girder is to be joined to point b of another girder. The active or output end of the fusing organ is placed in contact with points a and b. A stimulus into the input end of the fusing organ at time t causes points a and b to be welded together at time $t + 1$. The fusing organ can be withdrawn later. Connections may be broken by a *cutting organ* which, when stimulated, unsolders a connection.

The eighth part is a *muscle*, used to produce motion. A muscle is normally rigid. It may be connected to other parts. If stimulated at time t it will contract to length zero by time $t + 1$, keeping all its connections. It will remain contracted as long as it is stimulated. Presumably muscles can be used to move parts and make connections in the following way. Suppose that muscle 1 lies between point a of

[2] ["The General and Logical Theory of Automata." *Collected Works* 5.288–328. It will be recalled that the Illinois lectures were delivered in December of 1949.]

one girder and point *b* of another girder, and muscle 2 lies between point *a* and the active end *c* of a fusing organ. When both muscles are stimulated, they will contract, thereby bringing points *a*, *b*, and *c* together. When the fusing organ is stimulated, it will weld points *a* and *b* together. Finally, when the stimuli to the muscles are stopped, the muscles will return to their original length, at least one end of muscle 1 separating from the point *ab*. Von Neumann does not seem to have discussed the question of how the connections between muscles and other parts are made and broken.

Von Neumann conceived of an automaton constructing other automata in the following manner. The constructing automaton floats on a surface, surrounded by an unlimited supply of parts. The constructing automaton contains in its memory a description of the automaton to be constructed. Operating under the direction of this description, it picks up the parts it needs and assembles them into the desired automaton. To do this, it must contain a device which catches and identifies the parts that come in contact with it. The June, 1948 lectures contain only a few remarks on how this device might operate. Two stimulus units protrude from the constructing automaton. When a part touches them tests can be made to see what kind of part it is. For example, a stimulus organ will transmit a signal; a girder will not. A muscle might be identified by determining that it contracts when stimulated.

Von Neumann intended to disregard the fuel and energy problem in his first design attempt. He planned to consider it later, perhaps by introducing a battery as an additional elementary part. Except for this addition, von Neumann's early model of self-reproduction deals with the geometrical-kinematic problems of movement, contact, positioning, fusing, and cutting, and ignores the truly mechanical and chemical questions of force and energy. Hence I call it his *kinematic model* of self-reproduction. This early model is to be contrasted with his later *cellular model* of self-reproduction, which is presented in Part II of the present work.

In his June, 1948 lectures von Neumann raised the question of whether kinematic self-reproduction requires three dimensions. He suspected that either three dimensions or a Riemann surface (multiply-connected plane) would be needed. We will see in Part II that only two dimensions are required for self-reproduction in von Neumann's cellular model. This is a strong indication that two dimensions are sufficient for kinematic self-reproduction.

We return now to the Illinois lectures. Von Neumann discussed the general design of a self-reproducing automaton. He said that it

is in principle possible to set up a machine shop which can make a copy of any machine, given enough time and raw materials. This shop would contain a machine tool B with the following powers. Given a pattern or object X, it would search over X and list its parts and their connections, thereby obtaining a description of X. Using this description, the tool B would then make a copy of X. "This is quite close to self-reproduction, because you can furnish B with itself."]

But it is easier, and for the ultimate purpose just as effective, not to construct an automaton which can copy any pattern or specimen given to it, but to construct an automaton which can produce an object starting from a logical description. In any conceivable method ever invented by man, an automaton which produces an object by copying a pattern will go first from the pattern to a description and then from the description to the object. It first abstracts what the thing is like, and then carries it out. It's therefore simpler not to extract from a real object its definition, but to start from the definition.

To proceed in this manner one must have axiomatic descriptions of automata. You see, I'm coming quite close to Turing's trick with universal automata, which also started with a general formal description of automata. If you take those dozen elements I referred to in a rather vague and general way and give exact descriptions of them (which could be done on two printed pages or less), you will have a formal language for describing automata unambiguously. Now any notation can be expressed as a binary notation, which can be recorded on a punched tape with a single channel. Hence any automaton description could be punched on a piece of tape. At first, it is better not to use a description of the pieces and how they fit together, but rather a description of the consecutive steps to be used in building the automaton.

[Von Neumann then showed how to construct a binary tape out of rigid elements. See Figure 2. A binary character is represented at each intersection of the basic chain; "one" is represented by an attached rigid element, "zero" by the absence of a side element. Writing and erasing are accomplished by adding and removing side elements.]

I have simplified unnecessarily, just because of a purely mathematical habit of trying to do things with a minimum of notation. Since I'm using a binary notation, all I'm attaching here is no side chain, or a one-step side chain. Existing languages and practical notations use more symbols than the binary system. There is no difficulty in using more symbols here; you simply attach more complex side chains. In fact, the very linearity of our logical notation is

completely unnecessary here. You could use more complicated looped chains, which would be perfectly good carriers for a code, but it would not be a linear code. There is reason to suspect that our predilection for linear codes, which have a simple, almost temporal sequence, is chiefly a literary habit, corresponding to our not particularly high level of combinatorial cleverness, and that a very efficient language would probably depart from linearity.[3]

There is no great difficulty in giving a complete axiomatic account of how to describe any conceivable automaton in a binary code. Any such description can then be represented by a chain of rigid elements like that of Figure 2. Given any automaton X, let $\phi(X)$ designate the chain which represents X. Once you have done this, you can design a universal machine tool A which, when furnished with such a chain $\phi(X)$, will take it and gradually consume it, at the same time building up the automaton X from the parts floating around freely in the surrounding milieu. All this design is laborious, but it is not difficult in principle, for it's a succession of steps in formal logics. It is not qualitatively different from the type of argumentation with which Turing constructed his universal automaton.

Another thing which one needs is this. I stated earlier that it might be quite complicated to construct a machine which will copy an automaton that is given it, and that it is preferable to proceed, not from original to copy, but from verbal description to copy. I would like to make one exception; I would like to be able to copy linear chains of rigid elements. Now this is very easy. For the real reason it is harder to copy an existing automaton than its description is that the existing automaton does not conform with our habit of linearity, its parts being connected with each other in all possible directions, and it's quite difficult just to check off the pieces that have already been described.[4] But it's not difficult to copy a linear chain of rigid elements. So I will assume that there exists an automaton B which has this property: If you provide B with a description of anything, it consumes it and produces two copies of this description.

Please consider that after I have described these two elementary steps, one may still hold the illusion that I have not broken the principle of the degeneracy of complication. It is still not true that, starting from something, I have made something more subtle and more

[3] [The programming language of flow diagrams, invented by von Neumann, is a possible example. See p. 13 of the Introduction to the present volume.]

[4] [Compare Sec. 1.6.3 of Part II, written about 3 years later. Here von Neumann gives a more fundamental reason for having the constructing automaton work from a description of an automaton rather than from the automaton itself.]

involved. The general constructive automaton A produces only X when a complete description of X is furnished it, and on any reasonable view of what constitutes complexity, this description of X is as complex as X itself. The general copying automaton B produces two copies of $\phi(X)$, but the juxtaposition of two copies of the same thing is in no sense of higher order than the thing itself. Furthermore, the extra unit B is required for this copying.

Now we can do the following thing. We can add a certain amount of control equipment C to the automaton $A + B$. The automaton C dominates both A and B, actuating them alternately according to the following pattern. The control C will first cause B to make two copies of $\phi(X)$. The control C will next cause A to construct X at the price of destroying one copy of $\phi(X)$. Finally, the control C will tie X and the remaining copy of $\phi(X)$ together and cut them loose from the complex $(A + B + C)$. At the end the entity $X + \phi(X)$ has been produced.

Now choose the aggregate $(A + B + C)$ for X. The automaton $(A + B + C) + \phi(A + B + C)$ will produce $(A + B + C) + \phi(A + B + C)$. Hence auto-reproduction has taken place.

[The details are as follows. We are given the universal constructor $(A + B + C)$, to which is attached a description of itself, $\phi(A + B + C)$. Thus the process of self-reproduction starts with $(A + B + C) + \phi(A + B + C)$. Control C directs B to copy the description twice; the result is $(A + B + C) + \phi(A + B + C) + \phi(A + B + C)$. Then C directs A to produce the automaton $A + B + C$ from one copy of the description; the result is $(A + B + C) + (A + B + C) + \phi(A + B + C)$. Finally, C ties the new automaton and its description together and cuts them loose. The final result consists of the two automata $(A + B + C)$ and $(A + B + C) + \phi(A + B + C)$. If B were to copy the description thrice, the process would start with one copy of $(A + B + C) + \phi(A + B + C)$ and terminate with two copies of this automaton. In this way, the universal constructor reproduces itself.]

This is not a vicious circle. It is quite true that I argued with a variable X first, describing what C is supposed to do, and then put something which involved C for X. But I defined A and B exactly, before I ever mentioned this particular X, and I defined C in terms which apply to any X. Therefore, in defining A, B, and C, I did not make use of what X is to be, and I am entitled later on to use an X which refers explicitly to A, B, and C. The process is not circular.

The general constructive automaton A has a certain creative ability, the ability to go from a description of an object to the object. Like-

wise, the general copying automaton B has the creative ability to go from an object to two copies of it. Neither of these automata, however, is self-reproductive. Moreover, the control automaton C is far from having any kind of creative or reproductive ability. All it can do is to stimulate two other organs so that they act in certain ways, tie certain things together, and cut these things loose from the original system. Yet the combination of the three automata A, B, and C is auto-reproductive. Thus you may break a self-reproductive system into parts whose functioning is necessary for the whole system to be self-reproductive, but which are not themselves self-reproductive.

You can do one more thing. Let X be $A + B + C + D$, where D is any automaton. Then $(A + B + C) + \phi(A + B + C + D)$ produces $(A + B + C + D) + \phi(A + B + C + D)$. In other words, our constructing automaton is now of such a nature that in its normal operation it produces another object D as well as making a copy of itself. This is the normal function of an auto-reproductive organism: it creates byproducts in addition to reproducing itself.

The system $(A + B + C + D)$ can undergo processes similar to the process of mutation. One of the difficulties in defining what one means by self-reproduction is that certain organizations, such as growing crystals, are self-reproductive by any naive definition of self-reproduction, yet nobody is willing to award them the distinction of being self-reproductive. A way around this difficulty is to say that self-reproduction includes the ability to undergo inheritable mutations as well as the ability to make another organism like the original.

Consider the situation with respect to the automaton $(A + B + C + D) + \phi(A + B + C + D)$. By a mutation I will simply mean a random change of one element anywhere. If an element is changed at random in one of the automata A, B, or C, the system will usually not completely reproduce itself. For example, if an element is changed in C, C may fail to stimulate A and B at the proper time, or it may fail to make the connections and disconnections which are required. Such a mutation is lethal.

If there is a change in the description $\phi(A + B + C + D)$, the system will produce, not itself, but a modification of itself. Whether the next generation can produce anything or not depends on where the change is. If the change is in A, B, or C, the next generation will be sterile. If the change occurs in D, the system with the mutation is exactly like the original system, except that D has been replaced by D'. This system can reproduce itself, but its by-product will be

D' rather than D. This is the normal pattern of an inheritable mutation.

So, while this system is exceedingly primitive, it has the trait of an inheritable mutation, even to the point that a mutation made at random is most probably lethal, but may be non-lethal and inheritable.

PART *Two*

The Theory of Automata: Construction, Reproduction, Homogeneity

EDITORIAL NOTE

[The editor's insertions, commentaries, explanations, summaries, and Chapter 5 are in brackets. The figures are at the end of the volume.

The reader who wishes a general view of the contents of this part should examine Sections 1.1.2.3, 1.3.3.5, 2.8.2, 2.8.3, 4.1.1, 4.3.1, and 5.3.]

Chapter 1

GENERAL CONSIDERATIONS

1.1 Introduction

1.1.1.1 The theory of automata. The *formalistic* study of *automata* is a subject lying in the intermediate area between logics, communication theory, and physiology. It implies abstractions that make it an imperfect entity when viewed exclusively from the point of view of any one of the three above disciplines—the imperfection being probably worst in the last mentioned instance. Nevertheless an assimilation of certain viewpoints from each one of these three disciplines seems to be necessary for a proper approach to that theory. Hence it will have to be viewed synoptically, from the combined point of view of all three, and will probably, in the end, be best regarded as a separate discipline in its own right.[1]

1.1.1.2 The constructive method and its limitations. The present paper deals with a particular and limited phase of the formalistic theory of automata. The decisive limitation is that we will establish certain existence theorems, without, however, being able to prove that the constructions on which they are based are in any sense *optimal*, or that the postulates that they use are in any sense *minimal*. These questions of optimality and minimality could presumably be treated only if the methods for the formation of invariant quantitative concepts, and for their measurement, their evaluation, and the like, had been much further evolved in this subject of automata, control, and organization, than they are at present. We believe that such a development is possible and to be expected, and that it will to an important extent follow the patterns and the concept formations of thermodynamics.[2] The methods that will be used in this paper con-

[1] [Von Neumann here referred to Wiener. See Wiener's *Cybernetics* and von Neumann's review of it.]

[2] Von Neumann, "The General and Logical Theory of Automata" and "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." [See also the Third and Fourth Lectures of Part I of the present volume.]

tribute, however, only very partially to the effort that is needed in that direction, and, at any rate, we will limit ourselves at this occasion to the establishing of certain existences (by suitable, ad hoc constructions) in the sense outlined above.

1.1.2.1 The main questions: (*A*)–(*E*). Within the above limitations, however, we will deal with problems that are rather central—at least for the initial phases of the subject. We will investigate automata under two important, and connected, aspects: those of logics and of construction. We can organize our considerations under the headings of five main questions:

(A) Logical universality. When is a class of automata logically universal, i.e., able to perform all those logical operations that are at all performable with finite (but arbitrarily extensive) means? Also, with what additional—variable, but in the essential respects standard—attachments is a single automaton logically universal?

(B) Constructibility. Can an automaton be constructed, i.e., assembled and built from appropriately defined "raw materials," by another automaton? Or, starting from the other end and extending the question, what class of automata can be constructed by one, suitably given, automaton? The variable, but essentially standard, attachments to the latter, in the sense of the second question of (A), may here be permitted.

(C) Construction-universality. Making the second question of (B) more specific, can any one, suitably given, automaton be construction-universal, i.e., be able to construct in the sense of question (B) (with suitable, but essentially standard, attachments) every other automaton?

(D) Self-reproduction. Narrowing question (C), can any automaton construct other automata that are exactly like it? Can it be made, in addition, to perform further tasks, e.g., also construct certain other, prescribed automata?

(E) Evolution. Combining questions (C) and (D), can the construction of automata by automata progress from simpler types to increasingly complicated types? Also, assuming some suitable definition of "efficiency," can this evolution go from less efficient to more efficient automata?

1.1.2.2 The nature of the answers to be obtained. The answer to question (A) is known.[3] We will establish affirmative answers to

[3] Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem." [See the discussion of Turing machines and universal Turing machines at p. 49 ff. above. The indefinitely extendible tape of a Turing machine is the "variable, but essentially standard, attachment" von Neumann referred to in questions (A) and (B) above.]

questions (B)–(D) as well.[4] An important limitation to the relevance of a similar answer to question (E) lies in the need for a more unambiguous formulation of the question, particularly of the meaning of "efficiency." In addition, we will be able to treat questions (A)–(E) in this sense with much more rigid determinations as to what constitutes an automaton, namely with the imposition of what is best described as a *crystalline regularity*. In fact, this further result would seem to be at least as essential and instructive as the ability to answer questions (A)–(D) {and, to some extent question (E); cf. above} affirmatively.

In the balance of Chapter 1 we carry out a heuristic and preliminary discussion of questions (A)–(E). In Chapter 2 we develop a specific model, within the terms of which we can and will deal in full detail and rigorously with questions (A)–(D). Chapter 3 contains the analysis of another more natural, but technically more refractory, model. Chapter 4 is devoted to further heuristic considerations, which are more conveniently made after (and to some extent presuppose) the detailed constructions of Chapters 2 and 3.

[*1.1.2.3 Von Neumann's models of self-reproduction.* The preceding paragraph gives the plan von Neumann had in mind when he wrote the present chapter. Unfortunately, von Neumann was only able to carry out his intention through part of the planned Chapter 2. To understand the plan, and the several references he makes to it in the balance of the present chapter, one must know something about the various models of self-reproduction he considered. We will describe these models briefly in the present subsection. Of necessity, much of what we say here is based on personal communications from people with whom von Neumann discussed his models of self-reproduction.

Altogether, von Neumann considered five models of self-reproduction. We will call these the kinematic model, the cellular model, the excitation-threshold-fatigue model, the continuous model, and the probabilistic model.

The *kinematic model* deals with the geometric-kinematic problems of movement, contact, positioning, fusing, and cutting, but ignores problems of force and energy. The primitive elements of the kinematic model are of the following kinds: logical (switch) and memory (delay) elements, which store and process information; girders, which provide structural rigidity; sensing elements, which sense objects in the environment; kinematic (muscle-like) elements, which move objects around; and joining (welding) and cutting elements, which connect

[4] Von Neumann, "The General and Logical Theory of Automata," *Collected Works* 5.315–318.

and disconnect elements. The kinematic model of self-reproduction is described in the Fifth Lecture of Part I of the present work. As indicated there, von Neumann was thinking about it at least by 1948.

Von Neumann's second model of self-reproduction is his *cellular model.* It was stimulated by S. M. Ulam, who suggested during a discussion of the kinematic model that a cellular framework would be more amenable to logical and mathematical treatment than the framework of the kinematic model.[5] In the cellular model, self-reproduction takes place in an indefinitely large space which is divided into cells, each cell containing the same finite automaton. Von Neumann spoke of this space as a "crystalline regularity," a "crystalline medium," a "granular structure," and as a "cellular structure." [6] We will use the term *cellular structure.*

There are many possible forms of cellular structure which may be used for self-reproduction. Von Neumann chose, for detailed development, an infinite array of square cells. Each cell contains the same 29-state finite automaton. Each cell communicates directly with its four contiguous neighbors with a delay of at least 1 unit of time. Von Neumann developed this model in a manuscript entitled "Theory of Automata: Construction, Reproduction, Homogeneity," which constitutes the present Part II of the present volume. In a letter to me, Mrs. Klara von Neumann said of her husband's manuscript: "I am quite positive that it was started by him in late September 1952 and that he continued working on it until sometime in mid late 1953." As far as I can tell, von Neumann did little or nothing with the manuscript after 1953.

The manuscript as left by von Neumann had two completed chapters and a long but incomplete third chapter. Chapter 1 of the manuscript is the present chapter. Chapter 2 of the manuscript states the transition rule governing the 29-state cellular system; this is Chapter 2 below. The incomplete Chapter 3 of the manuscript carries out the fundamental steps in the design of a cellular self-reproducing automaton; it appears below as Chapters 3 and 4. Von Neumann never completed the design of his cellular self-reproducing automaton; I indicate how to do this in Chapter 5 below.

Von Neumann's cellular model of self-reproduction should be compared with some work of Ulam on cellular automata. In his *A Collec-*

[5] [See footnote 17 of Sec. 1.3.1.2 below. In his "Random Processes and Transformations," presented in 1950, Ulam described a cellular framework briefly and stated that it had been considered by von Neumann and him.]

[6] [Moore, "Machine Models of Self-Reproduction," suggested the name "tessellation model."]

tion of Mathematical Problems, Ulam formulated a matrix problem arising out of the cellular model. In his "On Some Mathematical Problems Connected with Patterns of Growth of Figures" and "Electronic Computers and Scientific Research," Ulam studied the growth of figures in cellular automata with simple transition rules. He also studied the evolution of successive generations of individuals with simple properties, each generation producing its successor in accordance with a simple, but non-linear recursive transformation.

Under a covering letter dated October 28, 1952, von Neumann sent a copy of the present Chapter 1 to H. H. Goldstine. This letter elaborates the plan of Section 1.1.2.2 above.

> This is the introduction—or "Chapter 1"—that I promised you. It is tentative and incomplete in the following respects particularly:
>
> (1) It is mainly an introduction for "Chapter 2" which will deal with a model where every cell has about 30 states. It refers only very incompletely to "Chapter 3" in which a model with excitation-threshold-fatigue mechanisms alone will be discussed, and to "Chapter 4" where I hope to say something about a "continuous" rather than "crystalline" model. There, as far as I can now see, a system of non-linear partial differential equations, essentially of the diffusion type, will be used.
>
> (2) The write-up is still in a quite "unliterary" form, i.e., there are no footnotes (only their places are indicated), references, explanations of the motivation, origin of the ideas, etc.

It is clear that when von Neumann wrote the present Chapter 1 he had the following plan in mind. Chapter 2 was to contain a complete development of the cellular model of self-reproduction. Chapter 3 was to treat an excitation-threshold-fatigue model of self-reproduction. Finally, Chapter 4 was to discuss a continuous model of self-reproduction. Von Neumann finished the essential steps in the design of the cellular model and then stopped. Unfortunately, he never found time to finish the cellular model or write about the other two models.

Von Neumann delivered the Vanuxem Lectures at Princeton University on March 2 through 5, 1953. There were four lectures, entitled "Machines and Organisms." The fourth was devoted to self-reproduction; the kinematic model, the cellular model, the excitation-threshold-fatigue model, and the continuous model were all mentioned. Since he had already agreed to give the manuscript "Theory of Automata: Construction, Reproduction, Homogeneity" to the University of Illinois Press, von Neumann did not himself want to write up these lectures separately. Instead, it was arranged that John Kemeny should write an article based on these lectures and the first two chapters of the manuscript. This was published in 1955 under the title "Man Viewed as a Machine." Much of the material of the

first three Vanuxem Lectures appeared later in *The Computer and the Brain.*

The *excitation-threshold-fatigue model*[7] of self-reproduction was to be based on the cellular model. Each cell of the infinite structure of the cellular model contains a 29-state automaton. Von Neumann's idea was to construct this 29-state automaton out of a neuron-like element which had a fatigue mechanism as well as a threshold. Since fatigue plays an important role in the operation of neurons, an excitation-threshold-fatigue model would be closer to actual systems than the cellular model. Von Neumann never discussed how an idealized neuron with fatigue would work, but we can design one by combining what he said about idealized neurons without fatigue with his account of the absolute and relative refractory periods of an actual neuron (cf. pp. 44–48 above and *Collected Works* 5.375–376).

An idealized excitation-threshold-fatigue neuron has a designated threshold and a designated refractory period. The refractory period is divided into two parts, an absolute refractory period and a relative refractory period. If a neuron is not fatigued, it becomes excited whenever the number of active inputs equals or exceeds its threshold. When the neuron becomes excited two things happen: it emits an output signal after a specified delay, and the refractory period begins. The neuron cannot be excited at all during the absolute refractory period; it can be excited during the relative refractory period, but only if the number of active inputs equals or exceeds a threshold which is higher than the normal threshold.

When an excitation-threshold-fatigue neuron becomes excited, it must remember this fact for the length of the refractory period and use this information to prevent input stimuli from having their normal effect on itself. Hence this kind of neuron combines switching, delay of output, and an internal memory with feedback to control the effect of incoming signals. Such a device is, in fact, a small finite automaton, that is, a device with inputs and outputs and a finite number of internal states. In the fourth Vanuxem Lecture von Neumann suggested that a neuron with threshold 2 and fatigue period 6 might supply most of the states of the 29-state finite automaton needed in each cell of his cellular framework.

[7] [It should be noted that the phenomenon that von Neumann calls fatigue is more often called refractoriness. In this more common usage, fatigue is a phenomenon involving many refractory periods. The absolute refractory period of a neuron determines a maximum rate at which it can be fired. Repeated firing of a neuron at, or close to, this rate produces an increase in threshold, making it more difficult to fire the neuron. This increase in threshold is the phenomenon commonly called "fatigue."]

The fourth model of self-reproduction which von Neumann considered was a *continuous model.* He planned to base this on a system of non-linear partial differential equations of the type which govern diffusion processes in a fluid. Von Neumann had worked on non-linear partial differential equations, and wanted to use automata heuristically to solve theoretical problems about such equations (cf. pp. 33–35 above). In the case of the continuous model of self-reproduction, he planned to proceed in the reverse direction, using non-linear partial differential equations to solve a problem of automata theory: the logical and mathematical nature of the process of self-reproduction. This was part of von Neumann's general plan to employ the techniques and results of that branch of mathematics known as analysis to solve problems in automata theory (cf. pp. 25–28 above).

The physics, chemistry, biology, and logic of a self-reproducing system are very complex, involving a large number of factors; for example, mass, entropy, kinetic energy, reaction rates, concentration of enzymes and hormones, transport processes, coding, and control. All the essential properties of the self-reproducing system must be represented in the equations by functions or dependent variables. Von Neumann recognized that a system of simultaneous non-linear partial differential equations adequate to account for self-reproduction would be much more complex than the systems usually studied.

Von Neumann had been trained as a chemical engineer and was therefore familiar with complex chemical reactions. He had also applied mathematics to complex physical systems of various kinds. He probably thought of the differential equations of self-reproduction in connection with his proposed excitation-threshold-fatigue model of self-reproduction. Assume that the cellular model is reduced to the excitation-threshold-fatigue model. The task then becomes that of formulating the differential equations governing the excitation, threshold, and fatigue properties of a neuron. The following processes are involved in neural activity.[8] The neuron is stimulated by inputs from other neurons. When the aggregate of these inputs reaches the threshold of the neuron, it excites the neuron by triggering a flow of sodium ions from the outside to the inside of the cell body. The flow or diffusion of ions causes the cell body to become depolarized. This diffusion and depolarization is then transmitted down the axon and constitutes the firing of the neuron. The firing is followed by a diffusion of potassium ions from the inside of the neuron to the outside,

[8] [For a complete description see Eccles, *The Neurophysiological Basis of Mind.*]

which repolarizes the neuron. The chemical balance of sodium and potassium is restored still later.

It is clear from the preceding description of the excitation, threshold, and fatigue processes of the neuron that chemical diffusion plays a fundamental role in these processes. This explains why von Neumann chose partial differential equations of the diffusion type for his continuous model of self-reproduction. The reason for von Neumann's choice of non-linear, rather than linear, differential equations is also clear. The kinematic, cellular, and excitation-threshold-fatigue models all show that switching operations (e.g., threshold, negation) as well as control loops involving branching, feedback, and delay, are essential to the logical, informational, and organizational aspects of self-reproduction. To model these discrete phenomena in a continuous system it is necessary to use non-linear partial differential equations.

The preceding plan for constructing the continuous model starts with a discrete system and proceeds to a continuous system. The cellular model of self-reproduction is developed first, it is then reduced to the excitation-threshold-fatigue model, and finally, this model is described by non-linear partial differential equations. The reverse procedure is often followed in science, and von Neumann was, of course, familiar with it. One takes a continuous system, such as a fluid with shock waves in it, and approximates this system by dividing it up into discrete cells, treating everything in a cell as if it were in the same state. In this way, the differential equations of the continuous system are replaced by the difference equations of the discrete system. One may then solve the difference equations on a digital computer, and under appropriate conditions the solution will approximate the solution of the differential equations.

But whatever the order of inquiry, a system of differential equations and the corresponding difference equations represents essentially the same phenomena. The transition rule for the cellular model (Ch. 2 below) is the difference equation version of the system of partial differential equations of the continuous model. The design of the primary automaton which reproduces itself corresponds to the boundary conditions on these partial differential equations. Another way to view the contrast between the continuous and cellular models is in terms of the difference between an analog and a digital computer. An analog computer is a continuous system, and a digital computer is a discrete system. Thus von Neumann's continuous model of self-reproduction stands in the same relation to analog computers as his cellular model of self-reproduction stands to digital computers. In Section 12 of his "Probabilistic Logics and the Synthesis of Reliable Organisms from

Unreliable Components," he proposed a scheme for representing and processing digital information in an analog device. His continuous model of self-reproduction should be compared with this scheme.

Von Neumann's continuous model of self-reproduction should also be compared with some work of Turing. In "The Chemical Basis of Morphogenesis," Turing analyzed morphogenesis by solving differential equations which describe the interaction, generation, and diffusion of chemical substances. Turing confined himself almost entirely to linear differential equations, but he touched on non-linear differential equations.

Von Neumann had been interested in the applications of probability theory throughout his career; his work on the foundations of quantum mechanics and his theory of games are examples. When he became interested in automata, it was natural for him to apply probability theory here also. The Third Lecture of Part I of the present work is devoted to this subject. His "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components" is the first work on probabilistic automata, that is, automata in which the transitions between states are probabilistic rather than deterministic. Whenever he discussed self-reproduction, he mentioned mutations, which are random changes of elements (cf. p. 86 above and Sec. 1.7.4.2 below). In Section 1.1.2.1 above and Section 1.8 below he posed the problems of modeling evolutionary processes in the framework of automata theory, of quantizing natural selection, and of explaining how highly efficient, complex, powerful automata can evolve from inefficient, simple, weak automata. A complete solution to these problems would give us a *probabilistic model of self-reproduction and evolution.*[9]]

1.2 The Role of Logics—Question (A)

1.2.1 The logical operations—neurons. In evaluating question (A), one must obviously consider automata which possess organs that can express the essential propositions of logics and which need not possess any other organs. This can be done by using organs each of which possesses two stable states, corresponding to the basic truth values of *true* and *false* in logics. It is convenient to use a plausible physiological analogy and to designate these organs (whatever they are or are thought to be in reality) as *neurons*, and the two above states as *excited* and *quiescent*, respectively. It is also convenient to attach to these states digital (arithmetical) symbols, namely 1 and 0, respec-

[9] [For some related work, see J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," and "Concerning Efficient Adaptive Systems."]

tively.[10] The familiar structure of logics can then be conveyed to an automaton built from such organs by connecting them with *lines* representing the logical implications, and by introducing a separate *species* of basic organs, i.e., of neurons, for each basic logical operation.[11] In the usual propositional calculus these are *and*, *or*, and *not*, to be designated by $\cdot$, $+$, and $-$, respectively.[12] The lines which control the neuron's behavior, i.e., which represent the logical variables that enter into the basic logical operation or function to which the neuron corresponds, are its *inputs;* the lines through which this neuron expresses its resulting behavior, i.e., which represent the value of the logical function in question, are the *outputs*. These are usually the inputs of other neurons. Instead of attributing to a neuron several outputs, it is preferable to allow only one, and to *split* it afterwards into as many branches as necessary. The time-element in the functioning of a neuron is best expressed by stipulating that the state prescribed by the logical function corresponding to the neuron (i.e., the value of that function) is assumed a fixed delay time τ after the neurons that control this behavior have assumed their relevant states. That is, the *response* of a neuron (on its output line) occurs a fixed delay time τ after the *stimuli* (on its input lines). It is unnecessary to allow propagation delays along lines; i.e., an output may be instantaneously active wherever it is an input. It is simplest to assume that all relevant events take place at times t that are integer multiples of τ: $t = n\tau$, $n = 0, \pm 1, \pm 2, \cdots$. Next, τ may be chosen as the unit of time: $\tau = 1$, and so always $t = 0, \pm 1, \pm 2, \cdots$.

The basic neurons referred to above are shown in Figure 3. Their behavior is described by the following rules:

1. a, b are the input lines; c is the output line of this neuron.

2.1. The $+$ neuron is excited at time t if and only if either the neuron with output line a or the neuron with output line b is excited at time $t - 1$.

2.2. The $\cdot$ neuron is excited at time t if and only if both the neuron with output line a and the neuron with output line b are excited at time $t - 1$.

2.3. The $-$ neuron ["minus neuron"] is excited at time t if and only if the neuron with output line a is not excited (i.e., is quiescent) at time $t - 1$.

[10] Boolean algebra is then applicable.

[11] McCulloch and Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity."

[12] Von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," Sec. 4.

The time delay caused by the operation of each neuron guarantees the effective and constructive character of the logical system arrived at in this manner.[13] It is easily seen—in fact it is essentially inherent in the correspondence between the species of neurons introduced and the basic operations of logics (cf. above and Fig. 3)—that automata built from these organs can express all *propositional functions* in logics.[14] Beyond this, the inclusion of inductive processes, and more generally, of all processes that are permissible in finitistic logics, requires a deeper analysis.[15] It brings in one substantively new element: the need for an arbitrarily large (finite, but freely adjustable in size) memory. This ties in with question (B) and will be considered subsequently.

1.2.2 Neural vs. muscular functions. Question (A) involved merely logical determinations; therefore it required only (at least directly *only;* cf., however, the last remark in Sec. 1.2.1) organs with two states, true and false. These two states are adequately covered by the neural states of excitation and quiescence. Question (B), on the other hand, calls for the construction of automata by automata, and it necessitates therefore the introduction of organs with other than logical functions, namely with the kinematical or mechanical attributes that are necessary for the acquisition and combination of the organs that are to make up the automata under construction. To use a physiological simile, to the purely neural functions must be added at least the muscular functions.

At this point several alternatives open up.

1.3 The Basic Problems of Construction—Question (B)

1.3.1.1 The immediate treatment, involving geometry, kinematics, etc. The most immediate approach is this. The constituent organs are the neurons and lines necessitated by (A), plus such additional organs as (B) (i.e., the present discussion) will require. These constituent organs are to be conceived of as physical objects in actual space. Their acquisition and combination (including the establishing of rigid connections between them) must accordingly take place in actual space, i.e., 3-dimensional, Euclidean space. (Further variations on the dimensionality and geometrical character of the space are possible, but we

[13] McCulloch and Pitts, *op. cit.* Von Neumann, *op. cit.*, Secs. 2.1 and 3.3.

[14] McCulloch and Pitts, *op. cit.* Von Neumann, *op. cit.*, Sec. 3.

[15] McCulloch and Pitts, *op. cit.* Von Neumann, *op. cit.*, Secs. 3.3 and 5.1. [Von Neumann also mentioned Kleene. He probably intended to refer to the Rand Corporation version of "Representation of Events in Nerve Nets and Finite Automata."]

will not consider them at this point. Cf., however, the crystal-lattice discussions later on.) The constituent organs that are needed for the automaton construction must thus be found and acquired in space, they must be moved and brought into contact and fastened together in space, and all automata must be planned as true geometrical (and kinematical and mechanical) entities. The functions that were described above, rather symbolically, as muscular, will now be very nearly truly muscular, etc. Different degrees of abstraction are still possible; for example, one may or may not pay attention to the truly mechanical aspects of the matter (the forces involved, the energy absorbed or dissipated, etc.). But even the simplest approach, which disregards the above-mentioned properly mechanical aspects entirely requires quite complicated geometrical-kinematical considerations.[16] Yet, one cannot help feeling that these should be avoided in a first attempt like the present one: in this situation one ought to be able to concentrate all attention on the intrinsic, logical-combinatorial aspects of the study of automata. The use of the adjective *formalistic* at the beginning of Section 1.1.1.1 was intended to indicate such an approach—with, as far as feasible, an avoidance of the truly geometrical, kinematical, or mechanical complications. The propriety of this desideratum becomes even clearer if one continues the above list of avoidances, which progressed from geometry, to kinematics, to mechanics. Indeed, it can be continued (in the same spirit) to physics, to chemistry, and finally to the analysis of the specific physiological, physico-chemical structures. All these should come in later, successively, and about in the above order; but a first investigation might best avoid them all, even geometry and kinematics. (A certain amount of primitive geometry and vestigial kinematics will appear even so, as will be seen later.)

1.3.1.2 The non-geometrical treatment—structure of the vacuum. A more sophisticated approach, which goes far towards meeting the desiderata expressed above, is this.[17]

The need to use geometry (and kinematics) merely expresses the fact that even the vacuum (the sites not now occupied, but potentially occupiable, by the constituent organs of automata) has a structure. Now 3-dimensional Euclidean space represents (or, represents with an approximation that is sufficient in the situation envisaged) the actual "structure of the vacuum." Nevertheless, this structure involves a number of traits which are unnecessarily complicating.

[16] Von Neumann, "The General and Logical Theory of Automata," *Collected Works* 5.315–318. [See also the Fifth Lecture of Part I above.]

[17] [Von Neumann was going to refer to S. Ulam here. See Sec. 1.1.2.3 above.]

While these will have to be considered at a later stage, it is desirable to eliminate them in the first approach. We will accordingly try to do this.

1.3.2 Stationarity—quiescent vs. active states. The main complication that we wish to remove is the influence of kinematics, that is, the necessity of moving objects around. It is preferable to have stationary objects only, normally in a *quiescent* state, and to postulate a system which will, under suitably and precisely defined conditions, transfer them from the quiescent state into an *active* state—or into a particular one of several possible active states.

1.3.3.1 Discrete vs. continuous framework. Next, these stationary and normally quiescent objects could be thought of as discrete entities, or as (infinitesimal) elements of a continuously extended medium. In the first case we will have a *granular* or *cellular* structure, while in the second case we are led back to a continuous space, more or less like that of Euclidean geometry.

1.3.3.2 Homogeneity: discrete (crystalline) and continuous (Euclidean). We now make the simplifying, but also very restrictive assumption,[18] that this spatial or quasi-spatial substratum be homogeneous. That is, the granular structure of the first case must have crystalline symmetry,[19] while the continuous space of the second case must be Euclidean.[20] In both cases this degree of homogeneity falls short of absolute homogeneity. Indeed, we have only postulated the homogeneity of the spatial (or, more broadly, combinatorial) matrix which carries the (quiescent or active) objects referred to above—the basic organs—but not the homogeneity of the population of these objects. That is, in the first (discrete) case, we have not postulated that all the cells of the crystal behave according to the same rules; and in the second (continuous) case, we have not postulated that the continuous, space-filling medium be subject everywhere to the same rules. Depending on whether this is, or is not, postulated, we will say that the system possesses, or does not possess, *intrinsic*, or *functional*, *homogeneity*.[21]

[18] Calling for stronger results. [That is, it is more difficult to construct a self-reproducing automaton in a homogeneous medium than in an inhomogeneous medium.]

[19] [A crystal is a solid body having a regular internal structure and bounded by symmetrically arranged plane surfaces intersecting at definite and characteristic angles. The regular internal structure consists of the rows and patterns of atoms of the crystal. The faces of the crystal express this regular internal structure externally.]

[20] [Apparently von Neumann was going to explain in this footnote why he was excluding the non-Euclidean spaces of Bolyai-Lobachevski and Riemann.]

[21] [In the discrete (crystalline, granular, cellular) case, functional homoge-

Beyond this, an even more complete homogeneity would exist if all (discrete or infinitesimal) elements were in the same state, for example, in the quiescent state (cf. above). In this case we will speak of *total homogeneity* or of *total quiescence*, respectively. This can obviously not be postulated in general, since it would exclude any positive and organized functioning of the automaton, for example, all moves in the sense of questions (B)–(E) of Section 1.1.2.1 above. It is, however, quite reasonable to strive to assume total quiescence as the initial state of an automaton. This cannot be absolutely enforced, since with the usual systems of rules total quiescence is a self-perpetuating state (cf. later). It will, however, be practical to assume a totally quiescent initial state and to proceed from there with the injection of a minimum amount of external stimulation (cf. later).

1.3.3.3 Questions of structure: (*P*)–(*R*). The point made at the beginning of Section 1.3.3.2, namely, that the homogeneity assumptions of Section 1.3.3.2 are seriously restrictive, is worth elaborating somewhat further. Indeed, even the manner in which the basic organs of the discussion of question (A) (the neurons of Section 1.2.1) are ordinarily strung together (cf. later), violates the first principle of homogeneity formulated in Section 1.3.3.2, that of the underlying granular structure, i.e., its crystalline symmetry. This degree of homogeneity can, however, be attained by some rather obvious and simple tricks, as will be seen later.

The systems that are thus obtained will, however, still violate the further, stricter, principle of homogeneity formulated in Section 1.3.3.2, that of functional homogeneity. This is clearly so, as long as several species of neurons are used (cf. Sec. 1.2.1, particularly Fig. 3) and these have to be distributed over the crystal lattice in an irregular (i.e., not crystalline-symmetric) manner. However, this use of several species of neurons and this distribution of them in an irregular manner are natural if one approaches the problem in the way which is the obvious one from the point of view of the ordinary logical and combinatorial techniques (cf. later). We will see that this difficulty, too, can be overcome and that functional homogeneity can be achieved. This, however, is considerably less easy, and it constitutes, in fact, one of the main results of this paper.

The homogeneity problem also raises some ancillary questions, which will successively occupy us when their respective turns come. They are the following:

---

neity means that each cell is occupied by the same finite automaton and each such automaton is connected to its neighbors in the same way. The particular cellular structure which von Neumann adopts in Ch. 2 is functionally homogeneous; see Sec. 1.3.3.5 below.]

(P) Which is the minimum usable dimensionality? (This question arises both in the first—discrete, crystalline—and in the second —continuous, Euclidean—cases of Secs. 1.3.3.1 and 1.3.3.2.)

(Q) In connection with functional homogeneity, can we require isotropy[22] in addition to homogeneity? {In the crystalline case this is meaningful for the regular crystal class only.[23] Cf. also question (R).}

(R) In the crystalline case, which crystal classes are usable? Or, to introduce our real aim explicitly, what is the highest degree of regularity that can be used?[23]

With respect to question (P), one would expect that the dimensionality 3 is usable and probably minimal. In fact even 2 is usable, while it seems unlikely that 1 should be, at least in combination with any otherwise plausible postulates.[24] These questions will be considered in some detail later.

Question (Q) will be considered later; it will appear that isotropy can be achieved, although non-isotropic models, too, can be of considerable interest.

As to question (R), we will use the maximum regularity, which is reasonably interpreted as the (body-centered) cubic class in 3 dimensions, and the corresponding quadratic class in 2 dimensions, the emphasis being, in view of what was said about question (P) above, on the latter. Some other classes, however, are also of interest, as will be seen later.

1.3.3.4 Nature of results, crystalline vs. Euclidean: statements (*X*)–(*Z*). To conclude this group of observations, we note this. We surmise that the comparative study of the two cases of Section 1.3.3.1 {i.e., of the crystalline (discrete) and of the continuous (Euclidean) case} will prove very rewarding. The available indications point strongly towards these conclusions:

[22] [A substance or space is isotropic insofar as it has the same properties in all directions. In the crystalline (discrete) case, functional isotropy means that each cell is connected to each of its immediate neighbors in the same way. The particular cellular structure which von Neumann adopts in Ch. 2 is functionally isotropic; see Sec. 1.3.3.5 below.]

[23] Crystal classes in two and three dimensions. [Crystals are divided into six groups or systems according to the number and nature of the axes of symmetry. Crystals belonging to the cubic system (also called "the isometric system" and "the regular system") are the most symmetrical of all crystals. In a crystal of the cubic system the three crystallographic axes of reference are at right angles to each other and are equal in length. Crystals having a cubic cell and crystals having an octahedral cell are the simplest of the forms of this class.]

[24] [Von Neumann was going to make a reference to Julian Bigelow and H. H. Goldstine here. They suggested modeling self-reproduction in 2 rather than 3 dimensions.]

(X) The general possibilities are about the same in the two cases.
(Y) The continuous case is mathematically much more difficult than the crystalline case.
(Z) If and when the appropriate analytical methods to deal with the continuous case are developed, this case will be more satisfactory and more broadly and relevantly applicable than the crystalline case.

These matters will be discussed in somewhat more detail later. We will make now only one more observation, which relates to the difficulties referred to in conclusions (Y) and (Z). These difficulties are due to the fact that, in the case in question, the mathematical problem becomes one of a system of non-linear, partial differential equations. It may be of some significance that non-linear partial differential equations, which in many directions define and limit our mathematical horizon, should make their appearance in this context also.

The difficulties referred to in (Y) and (Z) will cause us to direct our attention primarily to the crystalline case. In fact, we will from now on always have this case in mind, except where the opposite is expressly stated.

[*1.3.3.5 Homogeneity, quiescence, and self-reproduction.* In writing the present Part II of this volume, von Neumann was going through a process of reasoning which was to terminate in his cellular model of self-reproduction. At these early stages of the process he was exploring various possibilities, leaving specific choices until later. As the inquiry proceeded his terminology necessarily changed somewhat. Thus the "quiescence" of Section 1.3.3.2 above divides into "unexcitability" and "quiescence of an excitable cell" in Sections 1.3.4.1 and 1.3.4.2 below. A brief preview of the final outcome may help the reader to follow the development of von Neumann's thought.

Von Neumann's cellular structure is described in detail in Chapter 2 below. He chose an infinite 2-dimensional array of square cells. Each cell is occupied by the same 29-state finite automaton, and each such automaton is connected to its four immediate neighbors in exactly the same way; that is, the transition rule of Chapter 2 below is the same for each cell. Hence this cellular structure is "functionally homogeneous" in the sense of Section 1.3.3.2 above. Since each 29-state automaton is connected to each of its four neighbors in the same way, this structure is also isotropic in the sense of Section 1.3.3.3 above. Functional homogeneity and isotropy have only to do with structure, however, and not with content or state. Consequently, if different cells of a region of the cellular structure are in different states, one part of the region may act in one way and send informa-

tion in one direction, while another part of the region acts in a different way and sends information in a different direction.

The 29 states each cell is capable of assuming fall into three categories: unexcitable (1), excitable (20), and sensitized (8). These are listed later in Figure 9.

The unexcitable state **U** is utterly quiescent. This state plays a fundamental role with respect to the information content of the cellular structure, for this structure is operated in such a way that at each moment of time only a finite number of cells of the structure are in some state other than the unexcitable state. In this respect, the unexcitable state of von Neumann's system is analogous to a blank square on the tape of a Turing machine. Indeed, von Neumann represented zero in his linear array **L** by the unexcitable state **U**; see Section 1.4.2.5 below.

The 20 excitable states fall into three classes. There are 4 confluent states $\mathbf{C}_{\epsilon\epsilon'}$, where ϵ and ϵ' range over 0 and 1; the value "0" symbolizes quiescence, and the value "1" symbolizes excitation. There are 8 ordinary transmission states $\mathbf{T}_{0\alpha\epsilon}$ and 8 special transmission states $\mathbf{T}_{1\alpha\epsilon}$, where $\alpha = 0, 1, 2, 3$ and $\epsilon = 0, 1$ as before. Eight transmission states are quiescent and 8 are excited. The sensitized states are transient in nature, each lasting exactly 1 moment of time.

The set of 10 states consisting of the unexcitable state **U** and the quiescent states $\mathbf{C}_{00}$, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$) has this property: if every cell of the infinite cellular structure is in 1 of these 10 states, the system will never change (i.e., no cell will ever change state). The difference between the unexcitable state **U** and the 9 quiescent (but excitable) states $\mathbf{C}_{00}$, $\mathbf{T}_{u\alpha 0}$ is in the way they respond to stimuli (excitations). Stimuli entering a cell which is in the unexcitable state **U** convert that cell into 1 of the 9 quiescent states $\mathbf{C}_{00}$, $\mathbf{T}_{u\alpha 0}$. This conversion is the direct process of Section 2.6 below. The direct process takes 4 or 5 units of time, the sensitized states serving as intermediaries in the process.

A stimulus entering a cell in 1 of the 20 excitable states $\mathbf{C}_{\epsilon\epsilon'}$, $\mathbf{T}_{u\alpha\epsilon}$ ($\epsilon, \epsilon', u = 0, 1$; $\alpha = 0, 1, 2, 3$) does one of two things. It may change the cell back to the unexcitable state **U**; this is the reverse process of Section 2.5 below. Alternatively, this stimulus may be switched, combined with other stimuli, and delayed in the usual way. In particular, a quiescent finite automaton may be embedded in an area $\mathfrak{A}$ of the cellular structure by putting each of the cells of $\mathfrak{A}$ in 1 of the 10 states **U**, $\mathbf{C}_{00}$, and $\mathbf{T}_{u\alpha 0}$. If such a finite automaton is appropriately designed, it will compute in the usual way when it is stimulated (activated).

The temporal reference frame for the cellular structure consists of the times $\cdots -3, -2, -1, 0, 1, 2, 3, \cdots$. Von Neumann did not say exactly how he planned to use this temporal reference frame, but the following is consistent with what he said. All cells are in the unexcitable state for negative times. An *initial cell assignment* is a *finite* list of cells together with an assignment of a state to each cell of the list. At time zero an initial cell assignment is imposed on the cellular structure from the "outside," all cells not in the assignment list being left in the unexcitable state. Thereafter the cellular system runs according to the transition rule of Chapter 2. Each initial cell arrangement determines a unique history of the infinite cellular structure. We will call the infinite cellular structure together with an initial cell assignment an *infinite cellular automaton.*

An infinite cellular automaton which models self-reproduction operates as follows. The finite automaton **E** of Section 1.6.1.2 below constitutes an initial cell assignment. More specifically, the initial or starting state of this finite automaton **E** is an initial cell assignment. We impose this initial cell assignment on the cellular structure at time zero, thereby embedding the finite automaton **E** in the cellular structure. Let $\mathfrak{A}$ be the area of cells affected, so that initially all cells outside $\mathfrak{A}$ are unexcitable. The logical structure of **E** is such that at some later time τ another copy of **E** will appear in another area $\mathfrak{A}'$ of the cellular structure. That is, the state of each cell of $\mathfrak{A}'$ at time τ is identical to the state of the corresponding cell of $\mathfrak{A}$ at time zero. Thus **E** reproduces itself in area $\mathfrak{A}'$. In summary, this infinite cellular automaton has only one copy of **E** embedded in it at time zero, but two copies of **E** embedded in it at time τ. This is self-reproduction.

Let us look at the temporal development of an infinite cellular automaton. At negative times it is totally homogeneous in the sense of Section 1.3.3.2 above, all cells being unexcitable. At time zero this total homogeneity is modified by the introduction of inhomogeneity in a finite area. This inhomogeneity will, in general, propagate into surrounding areas. In the case of self-reproduction, the inhomogeneity of area $\mathfrak{A}$ spreads until area $\mathfrak{A}'$ is organized in the same way as area $\mathfrak{A}$.

The above treatment of infinite cellular automata makes no essential use of negative times. Since all cells are unexcitable at these times we could use only the times $0, 1, 2, 3, \cdots$ without any loss of generality. Von Neumann did not say why he introduced negative times. It is possible that he planned to use them in connection with a probabilistic model of self-reproduction and evolution (see the end of Sec. 1.1.2.3 above).]

1.3.4.1 Simplification of the problems of construction by the treatment

according to Section 1.3.1.2. We can now return to the original thought of Sections 1.2.2–1.3.3.1, that is, to the necessity of having organs that perform analogs of the muscular rather than of the neural function. In other words, we need organs which are concerned with the acquiring, positioning, and connecting of the basic organs of the automata under construction, rather than with purely logical operations in the sense of Section 1.2.1. Since the properly kinematic aspects of "acquiring" and "positioning" have been removed by the observations of Sections 1.3.1.2 and 1.3.2, the nature of the function, referred to above as an "analog of the muscular," must now be reconsidered.

The remark at the end of Section 1.3.2 makes it clear that this function now appears under the aspect of causing an object—or, to use a terminology suggested by Section 1.3.3.1, a *cell*—which is in a *quiescent* state, to go over into a suitable *active* state. Now the logical functions, as discussed in Section 1.2.1, also do this, but there is a difference here or, at least, a possibility of a difference. The nature of this difference can be best illustrated by a physiological simile.

1.3.4.2 Quiescence vs. activity; excitability vs. unexcitability; ordinary and special stimuli. A neuron may be quiescent or active, but it is at any rate potentially active; that is, it is an *excitable* cell. Connective tissue, on the other hand, consists of *unexcitable*, truly passive cells. So far the difference between an excitable, but (momentarily) quiescent cell, and a (permanently) passive cell is obvious. Now let us for a moment introduce the fiction that the growth of neurons (i.e., of excitable cells) occurs not by the formation of new cells, but by the transformation of existing, unexcitable cells into excitable ones. It should be noted that, while this is not so in reality, it is the arrangement that fits best into the picture of stationary cells introduced in Section 1.3.2. To reconcile this with reality, one may have to interpret the absence of a cell as the presence of one in a special, particularly unexcitable state. This concept is in reasonable harmony with the one relating to a "structure of the vacuum," as used in Section 1.3.1.2.

Such a transformation must itself be induced by some special stimuli, i.e., by some special active states of neighboring cells. The ordinary stimuli (i.e., the ordinary active states) which control the logical functions discussed in Section 1.2.1 cannot do this. These stimuli control transitions between quiescent and ordinary active states, but in excitable cells only, and without ever changing the species (in the sense of Sec. 1.2.1) of the cell (neuron) in question. Indeed it was just with respect to these ordinary stimulations that

unexcitability was defined above. In order to provide an equivalent of growth, the special stimulations referred to above must be able to cause transitions from unexcitability to excitability, and also to determine the specific species (in the sense of Sec. 1.2.1) of the excitable cell (neuron) thus created.

These concepts permit us to define the difference between quiescence (with excitability) and unexcitability; the former responds to (i.e., is removed by) ordinary stimuli, the latter only to special ones. This has, of course, only shifted the distinction into one between *ordinary* and *special* stimuli, that is, into a distinction between ordinary and special active states. Leaving the physiological simile, and returning to the mathematical problem at hand, the matter still presents some dubious aspects.

1.3.4.3 Critique of the distinctions of Section 1.3.4.2. Indeed, one must now consider critically the usefulness of the distinction between ordinary and special stimuli. As outlined above, the underlying idea is this. Ordinary stimuli are to be used for logical operations, taking the species of the neurons that are involved as fixed; that is, ordinary stimuli are to be used for the control and utilization of already satisfactorily organized sub-assemblies. Special stimuli are to be used for growth operations, involving the introduction of excitability, together with a new determination of the neuronic species, into previously unexcitable, or otherwise different areas. In other words, special stimuli are used for the organization (according to some logically determined plan) of hitherto unorganized (or differently organized) areas.

This distinction is certainly convenient for a first approach, since it permits one to keep conceptually fairly distinct functions quite sharply distinct in their actual embodiment and performance. We will therefore adhere to it strictly in our first constructions (cf. later). However, it is quite possible to relax it by various logical and combinatorial devices to varying degrees, up to (and including) complete obliteration. These turn out subsequently to be quite desirable for various mathematical and conceptual reasons, and we will therefore introduce them in our later constructions (cf. later).

[Logical functions and growth functions are fairly distinct conceptually. In his preliminary discussion von Neumann made a sharp distinction between their respective representations in his cellular system. Logical functions are performed by ordinary stimuli, and growth functions are performed by special stimuli. Later he relaxed the distinction somewhat by using both types of stimuli for both types of functions.

The final distinction between ordinary and special stimuli is shown in Figure 9. Both ordinary and special transmission stimuli bring about "growth" from the unexcitable state **U** to one of the nine quiescent states $\mathbf{C}_{00}$, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$); this is the direct process of Section 2.6 and Figure 10 below. Special transmission stimuli change an ordinary transmission state $\mathbf{T}_{0\alpha\epsilon}$ ($\alpha = 0, 1, 2, 3$; $\epsilon = 0, 1$) or a confluent state $\mathbf{C}_{\epsilon\epsilon'}$ ($\epsilon, \epsilon' = 0, 1$) into the unexcitable state **U**, while ordinary transmission stimuli change a special transmission state $\mathbf{T}_{1\alpha\epsilon}$ ($\alpha = 0, 1, 2, 3$; $\epsilon = 0, 1$) into the unexcitable state **U**; this is the reverse process of Section 2.5 below.

The logical functions of disjunction, conjunction, and delay will normally be performed by arrays of ordinary transmission and confluent states. The logical function of negation is not directly represented in von Neumann's system. Instead, negation will be accomplished by breaking a communication path and later restoring it. The breaking will be done by the reverse process and the restoring by the direct process. An example is given in Figure 17 of Section 3.2 below.]

1.4 General Construction Schemes—Question (B) Continued

1.4.1.1 Construction of cell aggregates—the built-in plan. The discussion up to this point (i.e., in Secs. 1.2.2–1.3.4.3) dealt only with the first part of question (B), the immediate problems of the construction of one automaton by another automaton. We can now pass to the second part of question (B), i.e., consider by what means a single automaton can be made to construct broad classes of other automata, and how variable, but essentially standard, attachments can be used to facilitate and extend this process.

Our discussion dealt so far only with the question: By what means can a single cell of specified characteristics be created? In this respect we developed some orienting principles. There remains the question of how this operation is to be controlled in all its details. It is clear that this will have to be done by the logical section of the primary (parent) automaton, which was considered in Section 1.2.1. It is also natural that this logical section of the primary automaton must supervise and sequence the multiplicity of acts of single-cell creation, which are necessary to produce the complete secondary (constructed) automaton.

This "sequencing" of the single cell creations has to be controlled by a logical pattern which is already laid out in the logical section of the primary automaton. Such a "logical pattern" is obviously neither more nor less than the complete "plan" of the secondary automaton

—functionally laid out within the primary automaton in "terms" that the primary automaton can "understand" and act on.

Thus the plan of the secondary automaton must be "built into" the primary automaton, presumably in terms of logical connections in the sense of Section 1.2.1.

1.4.1.2 The three schemes for building in multiple plans—the parametric form. The conclusion of Section 1.4.1.1 is that a primary automaton, constructed for this purpose, is prima facie (i.e., assuming the simplest type of design, which presents itself most immediately) suited to construct one and only one secondary automaton. Generalizations beyond this level are, however, immediate.

First, it is, of course, possible to have the plans of several (different) secondary automata built into the primary. Second, it is possible to incorporate the logical facilities that will make the primary automaton construct a specific secondary several times, e.g., a certain, preassigned number of times. Third, the plan of the secondary may contain a number of numerical parameters; and this plan can be built into the primary in this (variable) parametric form, together with facilities that make it possible to substitute any desired numerical values for these parameters.

The third scheme—or, rather, the combination of the second and the third schemes—is the most general of these. In their immediate form, however, these still contain a limitation, that is, a limitation of the numbers that can be used for the parameter values (third scheme) and for the number of repetitions (second scheme). Indeed, these numbers must be present in some form in the interior of the primary automaton, say in a digital representation. Assume that p such numbers are involved, and that they are all integers $\geqq 0$, say, $\nu_1, \cdots, \nu_p$. Let each one of those cells, which are to be used for their representation, have k states available for this purpose. It is best to interpret these states as the base k digits $0, 1, \cdots, k-1$. Let n_i such cells be available for ν_i, where $i = 1, \cdots, p$; this requires a total of $n = n_1 + \cdots + n_p$ cells. ν_i is thus expressed by n_i digits in a base k digital notation; hence it is limited to the k^{n_i} values $0, 1, \cdots, k^{n_i} - 1$.

1.4.2.1 The descriptive statement **L** *for numerical parameters.* The limitation just described can be circumvented by a simple trick: let these cells lie "outside" (i.e., not within the area of the primary automaton, but next to it), in the external, otherwise quiescent, region of the crystal. They might, for example, form a linear array **L** extending in the right hand (i.e., positive x) direction away from the area of the primary automaton. The k states used for these "notational" purposes must, of course, also be of a quasi-quiescent charac-

ter, i.e., such that they will normally not disturb (stimulate or otherwise transform) each other or the surrounding quiescent cells. This, however, is a desideratum that is easy to meet (cf. later). The primary automaton must then be endowed with the ability to establish contact with all parts of this linear array **L**, and to have its operations controlled, in the desired sense, by the "notational" (i.e., base k digital) states of the cells of **L**. One might, at first sight, expect difficulties in trying to do this for all possible **L** (for all possible sizes of **L**, i.e., of n) with a fixed, limited primary automaton. All these difficulties can, however, be overcome by fairly straightforward methods, as will appear when this matter is considered in detail [See Sec. 1.4.2.5 below]. We will mention here only one of them.

One might think that the "exploration" of **L** is not possible without specifying—i.e., expressing within the primary automaton—the size of **L**, i.e., n. In fact, it would seem natural that p and all the $\nu_1, \cdots, \nu_p$, must be so specified. This would again limit p and $n_1, \cdots, n_p$, since the primary automaton is a fixed entity, and hence it would limit **L** and through it the numbers that it represents.

This difficulty can be removed as follows. Let each cell in **L** have k states for notational purposes as before; i.e., the states corresponding to the digits $0, 1, \cdots, k-1$, and two additional states to be called *comma* and *period*. (All of these states must be "quasi-quiescent" in the sense indicated above.) Within **L**, the numbers p and $\nu_1, \cdots, \nu_p$ consist only of cells in digital states. Now let **L** be lined up as follows (proceeding from left to right, i.e., in the positive x-direction). The digits of p, a comma, the digits of ν_1, a comma, $\cdots$, the digits of ν_p, a period. The primary automaton, in "exploring" **L**, can sense the comma and the period and thereby ascertain the sizes of p and of the $\nu_1, \cdots, \nu_p$ no matter what these are.

1.4.2.2 Applications of **L**. The linear array **L** of Section 1.4.2.1 is the variable, but essentially standard attachment mentioned in question (B). It is the simple addendum to the primary automaton, which, although essentially quiescent and possessed only of the most rudimentary structure, expands the active potentialities of that automaton substantially, as appeared in Section 1.4.2.1. The possibilities that are inherent in this device will, however, become really clear only after this. The main application in this sense will be described later [Secs. 1.5 and 1.6]. We will consider first a lesser application.

1.4.2.3 Use of **L** *as an unlimited memory for* (*A*). The last mentioned application relates to the attachments to a purely logical automaton, referred to in question (A).

The setup for purely logical functions (in the sense of (A), as dis-

cussed in Section 1.2.1) fails to be universal because of the absence of one constituent: an arbitrarily large memory which is finite but of adjustable size (cf. the end of Sec. 1.2.1). The linear array **L**, as described in Section 1.4.2.1, is just that. Hence **L**, with its ancillary observation, exploration, and construction facilities, provides the (variable, but in the essential respects standard) attachment to the logical automaton, which bridges the gap to logical universality, as indicated in question (A). It should be noted that the need for the facilities ancillary to **L**, referred to above, means that componentry which is normally called for in the primary (construction) automata of (B) must also be introduced into the (logical) automata of (A), if logical universality is an aim. For the details of all this, cf. later [Chapters 4 and 5].

1.4.2.4 Use of base two for **L**. One more remark about the cells of **L** is in order: we can choose $k = 2$, i.e., let all representations of numbers be base 2. Then each cell in **L** must have $k + 2 = 4$ states for the purposes now under consideration (cf. the discussion in Sec. 1.4.2.1). If it is now desired to keep the number of states for notational purposes at 2, this can still be achieved. It suffices to replace each cell of **L** by 2 cells, since a pair of 2-valued states allows $2^2 = 4$ combinations.

The digitalization and punctuation scheme for **L** meets all the requirements of Section 1.4.2.1. It is, however, not the only possible one. The following is an obvious variant. Keep the punctuation states (the comma and the period), as in Section 1.4.2.1. Instead of the two base 2 digital states, designated 0 and 1, use only one, designated 1. Designate a number ν (an integer $\geqq 0$), not by a sequence of 0's and 1's, which express its base 2 digital expansion, but simply by a sequence of ν 1's. This representation is a good deal longer than that of Section 1.4.2.1 (ν symbols instead of n, where n is the smallest integer with $2^n > \nu$, i.e., with $n > {}^2\log \nu$), but it is more simply defined, and more simply exploitable (in the sense of the ancillary functions referred to in Sec. 1.4.2.3). For a detailed consideration, cf. later.

[*1.4.2.5 The linear array* **L**. It may be helpful at this point to anticipate von Neumann's design of the mechanism for reading and writing on an arbitrary cell of the unlimited linear array **L**.

Let us begin with a Turing machine, which is a finite automaton connected to an indefinitely extendible or infinite tape. The tape is divided into squares, each of which contains any one of a finite number of characters (i.e., is in any one of a finite number of states). Let the basic alphabet consist of two characters: zero and one, repre-

sented by a blank and a mark, respectively. At any given time the finite automaton senses one square of the tape. It can change the contents of this square (make a mark or erase a mark already there) and move the tape one square to the left or right, so that at the next moment of time it senses an adjacent square. Thus the finite automaton can, in a finite amount of time, gain access to, read, and modify any square of the tape.

It is clear that accessibility of an arbitrary tape square is the important thing, and having the tape move is only a means to this end. Alternatively, we can have the tape stand still and the finite automaton move back and forth along it. Or, we can have both the finite automaton and the tape stand still, and let the finite automaton communicate to an arbitrary square x_n by means of a contractable and indefinitely extendible "wire." The finite automaton can sense and modify the state of square x_n through this wire. Then the finite automaton can extend the wire to square x_{n+1}, or contract it to square x_{n-1}.

This last procedure is the one von Neumann used in his cellular system. The details are given in Chapter 4 below. We will explain the basic idea in connection with Figure 37. The memory control **MC** is a finite cellular automaton occupying the area indicated in that figure. **L** is an infinite array of cells extending to the right. "Zero" is represented in cell x_n by the unexcitable state **U**, and "one" is represented by the quiescent but excitable state $\mathbf{T}_{030}$, which is an ordinary transmission state directed downward.

To read cell x_n, the memory control **MC** sends a sequence of stimuli around the connecting loop $\mathbf{C}_1$ in the direction of the arrows. This sequence passes through x_n without affecting its neighbors x_{n-1} and x_{n+1}, is modified according to the state of x_n, and returns to the memory control **MC** with a representation of the contents of x_n. The memory control **MC** then writes on x_n and either extends the loop $\mathbf{C}_1$ so that it passes through cell x_{n+1}, or contracts the loop $\mathbf{C}_1$ so that it passes through cell x_{n-1}. The timing loop $\mathbf{C}_2$ is used in this extension-contraction process and is extended (or contracted) along with loop $\mathbf{C}_1$.

There are finitely many basic characters to be represented on **L**, including the period and comma. These are represented by binary sequences of some length k, and each character is stored in k cells of **L**. Initially, we will place a finite sequence of characters on **L**, of which the last one, and only the last one, is a period. Now, the memory control **MC** can sense the period and move it to the right or left as it expands or contracts the information on **L**. Hence, though **MC** is

of finite, fixed capacity, there is no bound to the amount of information on **L** with which it can interact.]

1.5 Universal Construction Schemes—Question (C)

1.5.1 Use of **L** *for non-numerical (universal) parametrization.* The schemes of Section 1.4.2.1 (together with Sec. 1.4.1.2) introduced an important broadening of the class of secondary automata that are constructible by one, suitably given, (fixed) primary automaton, in the sense of the second question of (B). They do not, however, achieve immediately the construction universality that is the aim of question (C). We will now get this, too, by introducing one further variation into the methods of Section 1.4.2.1.

The class of secondary automata which can be constructed according to Section 1.4.2.1 by a single primary automaton is limited in this sense. (We disregard for a moment the influence of Sec. 1.4.1.2.) These (secondary) automata may represent a broad class, but they must nevertheless all be particular specimens from a common species; that is, their individual (construction) plans all derive from a common master plan in which certain available parameters are specifically numerically substituted. In other words, even though the specific plan of the secondary automaton must no longer be built into the primary automaton, nevertheless the underlying, generic plan—the plan that controls all the subordinate plans—must be built in there.

1.5.2 The universal type of plan. Consider an arbitrary (but specifically given) secondary automaton and the possible ways to describe it. The following is certainly an adequate one:

(a) Specify the (two) x and the (two) y coordinates of the four sides of a rectangular area in which the entire secondary automaton is to be contained. Let these coordinates be x_1 , y_1 , x_2 , and y_2 . These coordinates should be counted from an origin which is at a suitably designated point within the area of the primary automaton.

It is actually better to introduce the side lengths $\alpha = x_2 - x_1 + 1$, $\beta = y_2 - y_1 + 1$ (assuming $x_1 \leqq x_2$, $y_1 \leqq y_2$) of the rectangular area containing the secondary, and to use the numbers x_1 , y_1 , α, β.

(b) According to (a) above, each cell within the rectangle covering the secondary can be characterized by two coordinates i $(= 0, 1, \cdots, \alpha - 1), j$ $(= 0, 1, \cdots, \beta - 1)$. (To be precise, with respect to the system of coordinates used in (a) above, the coordinates of the cell i, j are $x_1 + i$, $y_1 + j$.) This gives, as it should, $\alpha\beta$ cells in the rectangle covering the secondary. Let $\mathfrak{L}$ be the number of states that each one of these cells can assume, using accordingly an index $\lambda = 0, 1, \cdots, \mathfrak{L} - 1$ to enumerate these states. Designate by λ_{ij} the

state of cell (i, j) which is desired (on the basis of the plan of the secondary automaton in question) for the moment when the construction of this automaton is just completed.

It is clear from (a) and (b) that the secondary automaton is completely characterized by the specification of the numbers x_1, y_1, α, β and λ_{ij} for all pairs $i = 0, 1, \cdots, \alpha - 1$; $j = 0, 1, \cdots, \beta - 1$.

Note that these numbers have the following ranges:

$$x_1, y_1 = 0, \pm 1, \pm 2 \cdots .$$
$$\alpha, \beta = 1, 2, \cdots .$$
$$\lambda_{ij} = 0, 1, \cdots, \mathfrak{L} - 1 \quad \text{for} \quad \begin{matrix} i = 0, 1, \cdots, \alpha - 1; \\ j = 0, 1, \cdots, \beta - 1. \end{matrix}$$

To conclude, x_1, y_1 are better represented by specifying their absolute values $|x_1|$, $|y_1|$ and two numbers ϵ, η:

$$\epsilon \begin{cases} = 0 & \text{for} \quad x_1 \geqq 0 | \\ = 1 & \text{for} \quad x_1 < 0 |, \end{cases} \qquad \eta \begin{cases} = 0 & \text{for} \quad y_1 \geqq 0 | \\ = 1 & \text{for} \quad y_1 < 0 |. \end{cases}$$

Thus the sequence (of integers $\geqq 0$)

$$(*) \begin{cases} \epsilon, \eta, |x_1|, |y_1|, \alpha, \beta, \\ \lambda_{ij} \quad \text{for} \quad i = 0, 1, \cdots, \alpha - 1; \qquad j = 0, 1, \cdots, \beta - 1 \\ (\text{the } \lambda_{ij} \text{ are to be thought of as lexicographically ordered by } i, j), \end{cases}$$

contains a complete description of the desired secondary automaton, in the condition—i.e., with the cell-states—actually desired for its initial moment, immediately after completion.

This sequence of numbers may now be treated with the method described in Section 1.4.2.1 for the simpler sequence that occurred there (the sequence p, ν_1, $\cdots$, ν_p). That is, we can form a linear array of cells **L**, extending in the right hand (i.e., positive x) direction, and made up as follows: the numbers enumerated in formula $(*)$ in the order in which they appear there, each represented by its base k digital expansion, any two consecutive ones separated by a comma, and the last one followed by a period. The general description above plays now precisely the role of the general plan of a class of secondary automata, which contains parameters, as described in connection with the third scheme in Section 1.4.1.2. In addition to this, the linear array **L** introduced above is the exact equivalent of the linear array **L** introduced in Section 1.4.2.1: it specifies the numerical values that have to be substituted for the parameters of the general description. Thus the present description of an arbitrary secondary automaton has been made to fit entirely into the parametrization pattern of the

third scheme of Section 1.4.1.2. Since it is entirely unrestricted, this means that the universality referred to in question (C) can be achieved in this way.

1.6 Self-Reproduction—Question (D)

1.6.1.1 The apparent difficulty of using **L** *in the case of self-reproduction.* Let us now consider question (D), that is, the problem of self-reproduction.

The a priori argument against the possibility of self-reproduction is that it is natural to expect the constructing automaton to be more complex than the constructed one—i.e., the primary will be more complex than the secondary.[25] This is confirmed by the results outlined in Sections 1.2.2–1.4.1.1, i.e., those dealing with the first question in (B): the primary must contain a complete plan of the secondary (cf. Sec. 1.4.1.1), and in this sense the primary is more complex than the secondary. This limitation is somewhat transformed, but not removed, by the subsequent developments of Sections 1.4.1.1–1.5.2; even the strongest one among the results that are discussed there (the answer to question (C) in Sec. 1.5.2, securing universality) is subject to one form of it. Indeed, this result calls for a complete description of the secondary, expressed by the linear array of cells **L**, to be attached to the primary.

If one tried to pass from here directly to self-reproduction, it would be necessary to have an automaton which can contain its own plan, for example, in the form **L**. If the second question of (D) is included, it would also have to contain the plan (i.e., the **L**) of another, prescribed automaton.

With the scheme of Section 1.5.2, even the first is impossible: the (secondary) automaton considered there has no more than $\alpha\beta$ cells, while **L** (according to formula (*) in Sec. 1.5.2) consists of $\alpha\beta + 6$ digitalized numbers, $\alpha\beta + 5$ commas, and a period (i.e., $2\alpha\beta + 12$ or more cells). Many variants on this theme are possible, but none has yet appeared which, when directly used, overcomes this difficulty. However, there is an indirect method that circumvents it.

1.6.1.2 Circumvention of the difficulty—the types **E** *and* $\mathbf{E_F}$. This method is as follows.[26]

Designate the universal (primary) automaton of Section 1.5.2 by **A**. **A** constructs any secondary whose description **L** is attached to **A**, as described in Section 1.5.2.

[25] [See also pp. 79–80 above.]

[26] Von Neumann, "The General and Logical Theory of Automata." [See also pp. 84–87 above.]

It is possible to design and to position at a definite place adjacent to **A** another automaton **B** with the following function. **B** explores **L** and produces an exact copy $\mathbf{L}'$ of it, placing $\mathbf{L}'$ in exactly the same position with respect to the secondary that **L** is in with respect to the primary **A**. The information necessary for this positioning can be obtained by the investigation of **L**, since the latter contains the numbers x_1, y_1, α, β, which describe the position of the secondary in question with respect to the primary **A**.

Consider finally an automaton **C** which controls the two previously mentioned automata **A** and **B** as follows: **C** first causes **A**, as primary, to build the secondary **S** described by **L**. **C** then causes **B** to make a copy $\mathbf{L}'$ of **L** and to attach it to the secondary **S** as described above. Now designate the total aggregate of all three automata **A**, **B**, **C** by **D**.

Designate the description **L** of this automaton **D** by $\mathbf{L_D}$. Note that $\mathbf{L_D}$ must contain the numbers x_1, y_1 (indirectly, by way of ϵ, η, $|x_1|$, $|y_1|$; cf. formula (*) in Sec. 1.5.2), α, β, which describe the positioning of the desired secondary with respect to the primary. There need be no doubt about the values of α, β that are to be used here, since one can ascertain how large a rectangle is needed to cover **D**. With respect to x_1, y_1, however, there is a real choice; these two coordinates define the relative position of the desired secondary with respect to the primary. Let us assume first that this choice will be made in some definite manner; it need only guarantee that the secondary and its attachment $\mathbf{L}'$ will lie wholly outside the primary and its attachment **L**. Later on we will have somewhat more to say about this.

Now consider the complex **E** which results from attaching $\mathbf{L_D}$ to **D**. By going over the description given above, it is easily verified that **E** will construct precisely **D** with $\mathbf{L_D}$, displaced as above. Thus **E** is self-reproducing.

This answers the first question of (D). The second question of (D) can now also be answered along the same lines. Indeed, assume that in addition to self-reproduction, the construction of a further automaton **F** is also wanted. In this case, form the **L** which describes **D** followed by **F**: $\mathbf{L_{D+F}}$. Now consider the complex $\mathbf{E_F}$ which results from attaching $\mathbf{L_{D+F}}$ to **D**. It is clear that this will construct **D**, attach $\mathbf{L_{D+F}}$ to it, and also construct **F**. In other words, it self-reproduces and constructs **F** in addition.

The following remarks serve to clarify somewhat further the nature of the procedure outlined for (D).

1.6.2.1 First remark: shape of **L**. The construction of an automaton was based on generating separately every correct state of every cell

in a suitable covering area {cf. Sec. 1.5.2 (b), where this modus procedendi is indicated}. The covering area is conceived in a simplified, and therefore presumably often overextended, form as a rectangle {cf. Sec. 1.5.2 (a)}. The external attachment **L** is a linear array (cf. the last part of Sec. 1.5.2). These two geometrical shapes will not always fit together perfectly: covering them simultaneously by a rectangle may force an inelegant overextension of the latter. It should be pointed out that there is nothing immutable about the linear shape of **L**, and that one might well decide to change it (cf. later). On the other hand, the linear shape has the virtue of easy overall accessibility (cf. later [Ch. 4]).

1.6.2.2 Second remark: avoidance of collision in a single reproduction. As pointed out toward the end of Section 1.6.1.2, x_1, y_1 must be so large that the secondary (whose position relative to the primary is defined by the coordinates x_1, y_1) and its attachment $\mathbf{L}'$ will lie wholly outside the primary and its attachment **L**. Hence they are affected by the size of **L**. ($\mathbf{L}'$ is congruent to **L**.) This creates the danger of a vicious circle, since **L** contains $|x_1|$, $|y_1|$.

However, this danger is not serious, and any one of the following procedures will obviate it.

L (both for the primary **L** itself and for the secondary $\mathbf{L}'$) extends in one direction only (the positive x-direction; cf. the end of Sec. 1.5.2), which implies that it is quite thin in the y-directions (especially if it is linear; cf. above and also later). Therefore, a fixed minimum value for $|y_1|$ can be assigned, which guarantees that neither **D** nor **L** of the primary and of the secondary collide, by virtue of their separation in the y-direction.

Alternatively, a base k notation for $|x_1|$, $|y_1|$ (cf. Secs. 1.4.1.2 and 1.4.2.1) guarantees that the area used for their designation, and therefore **L** also, increases only as the $^2\log$ of these numbers (cf. Sec. 1.4.2.4), whereas the separation that they provide is essentially that of their own size. Clearly for sufficiently large numbers these will overtake their own $^2\log$ to any desired degree.

Finally, if each number is to be designated, as alternatively suggested in Section 1.4.2.4, by a sequence of as many ones as it expresses, we can still avoid any difficulty by, for example, agreeing that $|x_1|$, $|y_1|$ are to be squares of integers and that the numbers to be designated by the means indicated above will be their square roots. Thus the required size of **L** will go with the square root of the separation provided, which is, like the $^2\log$, a slowly increasing function, sure to be adequately overtaken when the numbers get sufficiently large.

As mentioned above, any one of these three devices is workable,

and they are not the only ones. The actual procedure will be developed later.

1.6.2.3 Third remark: analysis of the method for overcoming the difficulty of Section 1.6.1.1—the role of **L**. It is worth recapitulating how the a priori argument against the possibility of self-reproduction, as stated in Section 1.6.1.1, was overcome in Section 1.6.1.2.

The essential step was that **D** contained a sub-assembly **B** which is able to copy (and re-position) any linear array **L**. **B** is a fixed entity, of fixed, finite size, and it is yet able to copy an **L** of any size. It is essentially this step of "copying" which transcends the otherwise seemingly valid rule of the primary's necessary superiority (in size, also in organization) over the secondary.

Now $\mathbf{L} = \mathbf{L}_{\mathbf{G}}$ is the description of the secondary **G** that is to be constructed, as discussed in Section 1.5.2. (In our actual applications in Sec. 1.6.1.2, **D** and $\mathbf{D} + \mathbf{F}$ played the role of **G**.) One might ask why the description $\mathbf{L}_{\mathbf{G}}$ is preferable to the original **G** in controlling the copying device **B**. In other words, why can **B** not copy directly **G** itself, i.e., why must the intermediary $\mathbf{L}_{\mathbf{G}}$ be introduced? This question is clearly of considerable semantic importance for the area in which we are now working, i.e., for a theory of automata. Indeed, it touches at the base of the entire question of notations and representations, i.e., of the significance and advantages of introducing "descriptions" in addition to the original objects.

The reason is this. In order to copy a group of cells according to the ideas of Section 1.6.1.2 concerning **B**, it is necessary to "explore" that group to ascertain the state of each one of its cells and to induce the same state in the corresponding cell in the area where the copy is to be placed. This exploration implies, of course, affecting each cell of this group successively with suitable stimuli and observing the reactions. This is clearly the way in which the copying automaton **B** can be expected to operate, i.e., to take the appropriate actions on the basis of what is found in each case. If the object under observation consists of "quasi-quiescent" cells (cf. the remarks made on this subject in Sec. 1.4.2.1), then these stimulations can be so arranged as to produce the reactions that **B** needs for its diagnostic purposes, but no reactions that will affect other parts of the area which has to be explored. If an assembly **G**, which may itself be an active automaton, were to be investigated by such methods, one would have to expect trouble. The stimulations conveyed to it, as discussed above, for "diagnostic" purposes, might actually stimulate various parts of **G** in such a manner that other regions could also get involved, i.e., have the states of their cells altered. Thus **G** would be disturbed;

it could change in ways that are difficult to foresee, and, in any case, likely to be incompatible with the purpose of observation; indeed, observing and copying presuppose an unchanging original. The virtue of $\mathbf{L_G}$ (as compared to **G**) is that, since it consists of quasi-quiescent cells, no such complications (i.e., no spreading of the diagnostic stimulations) need be expected. (For the details of all this, cf. later [Ch. 4].)

The above requires one more qualification. Our choice actually did not lie between the copying of **G** and the copying of $\mathbf{L_G}$. It was rather the copying of **G** on the one hand, and the copying of $\mathbf{L_G}$, combined with the construction of **G** from its description $\mathbf{L_G}$, on the other hand. The last step in the second procedure, however, is feasible, since this is precisely what the universal constructing automaton in the sense of question (C) will do, according to Section 1.5.2. Note also that the quasi-quiescent character of $\mathbf{L} = \mathbf{L_G}$ is important in this construction step too; in fact, the observations of Section 1.4.2.1 concerning quasi-quiescence in **L** were aimed directly at this application.

1.6.3.1 Copying: use of descriptions vs. originals. It is worthwhile to observe at this point, too, why a third step, namely the construction of $\mathbf{L_G}$, based on a direct exploration of the original **G**, cannot be carried out with these methods. Note that if this could be done, then a suitable primary automaton could copy a given automaton **G** without ever having been furnished with its description $\mathbf{L_G}$. Indeed, one would begin with the step mentioned above, the construction of $\mathbf{L_G}$ from **G**, and then proceed with the two steps mentioned previously, the copying of $\mathbf{L_G}$ and the construction of **G** from $\mathbf{L_G}$. The difficulty is that the two last mentioned steps require only the observation of the quasi-quiescent $\mathbf{L_G}$, while the first mentioned step would also call for the observation of the uncontrollably reactive **G**. If one considers the existing studies concerning the relationship of automata and logics, it appears very likely that any procedure for the direct copying of a given automaton **G**, without the possession of a description $\mathbf{L_G}$, will fail; otherwise one would probably get involved in logical antinomies of the Richard type.[27]

To sum up, the reason to operate with "descriptions" $\mathbf{L_G}$ instead of the "originals" **G** is that the former are quasi-quiescent (i.e., unchanging, not in an absolute sense, but for the purposes of the exploration that has to be undertaken), while the latter are live and reactive. In the situation in which we are finding ourselves here, the

[27] [Von Neumann indicated that he was going to make a footnote reference to Turing at this point. See Sec. 1.6.3.2 below.]

importance of descriptions is that they replace the varying and reactive originals by quiescent and (temporarily) unchanging semantic equivalents and thus permit copying. Copying, as we have seen above, is the decisive step which renders self-reproduction (or, more generally, reproduction without degeneration in size or level of organization) possible.

[*1.6.3.2 The Richard paradox and Turing machines.* As indicated above, von Neumann was going to make a footnote reference to Turing in connection with the Richard paradox. I do not know what he had in mind, but I think it likely that he was going to mention the parallelism between Richard's paradox[28] and Turing's proof of the undecidability of the halting problem. In any case, this parallelism is illuminating in the present context.

Richard's paradox may be generated in a suitable language £ as follows. Let e_0 , e_1 , e_2 , $\cdots$ be an enumeration of all the expressions of £ which define two-valued number-theoretic functions of one variable, that is, functions from the natural numbers to the two values zero and one. The expression "x is odd" is such an expression; it defines a function which is true (has the value 1) for odd numbers and is false (has the value 0) for even numbers. Let $f_i(n)$ be the number-theoretic function defined by e_i , and define $-f_i(n)$ by

$$-f_i(n) = 0 \text{ if } f_i(n) = 1$$
$$-f_i(n) = 1 \text{ if } f_i(n) = 0.$$

Finally, let e' be the expression "the function $-f_n(n)$."

We assume that e' is expressible in £, and derive a contradiction. (1) The enumeration e_0 , e_1 , e_2 , $\cdots$ contains all the expressions of £ which define two-valued number-theoretic functions of one variable. Expression e' clearly defines a two-valued number-theoretic function of one variable. Therefore expression e' is in the enumeration e_0 , e_1 , e_2 , $\cdots$. (2) But e' is an explicit definition of the function $-f_n(n)$, which differs from every function in the enumeration $f_0(n)$, $f_1(n)$, $f_2(n)$, $\cdots$. Therefore e' does not define any of the functions $f_0(n)$, $f_1(n)$, $f_2(n)$, $\cdots$. For each i, $f_i(n)$ is defined by e_i . Consequently, e' is not in the enumeration e_0 , e_1 , e_2 , $\cdots$.

Thus we have shown both that the expression e' is in the enumeration e_0 , e_1 , e_2 , $\cdots$ and that it is not in this enumeration. The appearance of this contradiction is surprising, because it would seem that expression e' is a legitimate expression in a consistent language, namely, the English language enriched with some mathematical

[28] [Richard, "Les principes des mathématiques et le problème des ensembles." See also Kleene, *Introduction to Metamathematics*, pp. 38, 341.]

symbols. Actually, the contradiction shows that if a language $\mathfrak{L}$ is consistent then e' cannot be expressed in it.

Let us turn next to the halting problem for Turing machines. This problem was explained at the end of the Second Lecture of Part I above. A Turing machine is a finite automaton with an indefinitely extendible tape. A "concrete Turing machine" is a Turing machine which has a finite "program" or problem statement on its tape initially. A concrete Turing machine is said to be "circular" if it prints a finite sequence of binary digits and halts, while it is said to be "circle-free" if it continues to print binary digits in alternate squares forever. Turing proved that there is no decision machine for halting, that is, no abstract Turing machine which can decide whether an arbitrary concrete Turing machine is circular (will halt sometime) or circle-free.

Turing's proof that there is no decision machine for halting may be put in a form which closely parallels the preceding proof concerning Richard's paradox. Let t_0, t_1, t_2, $\cdots$, t_i, $\cdots$ be an enumeration of all the circle-free concrete Turing machines. Let $s_i(0)$, $s_i(1)$, $s_i(2)$, $\cdots$, $s_i(n)$, $\cdots$ be the sequence computed by machine t_i. Each $s_i(n)$ is either zero or one, so machine t_i computes the two-valued function $s_i(n)$ in the sense of enumerating its values in their natural order. Now consider the function $-s_n(n)$. This function is analogous to the function $-f_n(n)$ defined by expression e' in the Richard paradox.

To continue the parallelism, we assume that there is a circle-free concrete Turing machine t' which computes the function $-s_n(n)$ and derive a contradiction. (1) The enumeration t_0, t_1, t_2, $\cdots$ contains all circle-free concrete Turing machines. Machine t' is by hypothesis a circle-free concrete Turing machine. Consequently, machine t' is in the enumeration t_0, t_1, t_2, $\cdots$. (2) By definition, t' computes the function $-s_n(n)$, which clearly differs from every function in the enumeration $s_0(n)$, $s_1(n)$, $s_2(n)$, $\cdots$. For each i, the function $s_i(n)$ is computed by the machine t_i. Consequently, machine t' is not in the enumeration t_0, t_1, t_2, $\cdots$.

Thus we have shown both that machine t' is in the enumeration t_0, t_1, t_2 $\cdots$ and that it is not. The appearance of this contradiction is not surprising, however, for we had no reason to believe that machine t' exists. In other words, the contradiction shows that machine t' does not exist, and hence that the function $-s_n(n)$ is not computed by any circle-free concrete Turing machine.

We assume next that there is a decision machine t^h for halting, and derive a contradiction. There is a concrete Turing machine which

can enumerate all the concrete Turing machines; call it t^e. The output of t^e can be fed into t^h to produce a machine $t^e + t^h$ which enumerates all the circle-free concrete Turing machines. There is an abstract Turing machine t^u which can simulate each circle-free concrete machine in turn, find $s_n(n)$ for each machine n, and print $-s_n(n)$. Thus the machine $t^e + t^h + t^u$ computes the function $-s_n(n)$, and is the machine t'. But we know from the preceding paragraph that machine t' does not exist. Machines t^e and t^u do exist. Therefore, machine t^h does not exist. That is, there is no decision machine for halting. It follows also that machine $t^e + t^h$ does not exist, i.e., there is no machine which enumerates all the circle-free concrete Turing machines.

The first part of the preceding proof that there is no decision machine for halting establishes both that machine t' is in the enumeration $t_0, t_1, t_2, \cdots$ and that it is not. This closely parallels the earlier proof, given in connection with Richard's paradox, that the expression e' is in the enumeration $e_0, e_1, e_2, \cdots$ and that it is not. Both use Cantor's diagonal procedure to define a function which is not in a given enumeration. I suspect that it was because of this parallelism that von Neumann was going to refer to Turing at this point.

It should be noted that the Richard paradox can be barred from a language by imposing a "theory of types" on that language.[29] For example, we can design the language so that every expression of the language has a type number, and so that an expression of given type can refer only to expressions of lower type. Suppose now that the expressions $e_0, e_1, e_2, \cdots$ are of type m. Since expression e' refers to all these expressions, it must be of higher type, and therefore cannot be in the list $e_0, e_1, e_2, \cdots$. This being so, our earlier derivation of Richard's paradox fails. See in this connection the letter from Kurt Gödel quoted at the end of the Second Lecture of Part I above.

These considerations about self-reference are relevant to the problem of designing a self-reproducing automaton, since such an automaton must be able to obtain a description of itself. In Section 1.6.3.1 (entitled "Copying: use of descriptions vs. originals") von Neumann considers two methods for accomplishing this, which I will call the "passive" and "active" methods. In the passive method the self-reproducing automaton contains within itself a passive description of itself and reads this description in such a way that the description

[29] [Russell, "Mathematical Logic as Based on the Theory of Types." See also Kleene, *Introduction to Metamathematics*, pp. 44–46.]

cannot interfere with the automaton's operations. In the active method the self-reproducing automaton examines itself and thereby constructs a description of itself. Von Neumann suggests that this second method would probably lead to paradoxes of the Richard type, and for this reason he adopts the first method. See also Sections 1.7.2.1, 2.3.3, 2.6.1, and 2.8.2 below. We will see by the end of Chapter 5 below that a self-reproducing machine can indeed be constructed by means of the first method. This shows that it is possible for an automaton to contain a description of itself.][30]

1.7 Various Problems of External Construction Intermediate Between Questions (D) and (E)

1.7.1 Positioning of primary, secondary, ternary, etc. We pass now to an extension of question (D) which points the way towards question (E). This deals with the question of positioning the secondary that the self-reproducing primary **E** or $\mathbf{E_F}$ constructs, and the initiation, timing, and repetitions of the act of self-reproduction.

Note that the positioning of **F** for $\mathbf{E_F}$ need not create any new problems: $\mathbf{E_F}$ is **D** with $\mathbf{L_{D+F}}$ attached (cf. the end of Sec. 1.6.1.2) and $\mathbf{L_{D+F}}$ is a description of **D** followed by a description of **F**. In this joint description of **D** with **F** the latter must be unambiguously positioned with respect to the former, and this takes care of what is needed in this respect.

Returning to the main question of positioning the secondary by **E** or $\mathbf{E_F}$, we can argue as follows. Assume that this positioning is done by the first method of Section 1.6.2.2, i.e., by choosing a $\bar{y}$ such that $|y_1| \geqq \bar{y}$ guarantees the separateness of the primary and the secondary **E** or $\mathbf{E_F}$. (In the case of $\mathbf{E_F}$ we think of both primary and secondary as provided with an **F** positioned according to $\mathbf{L_{D+F}}$ relatively to **D**; cf. above, although at the beginning of the process only the secondary need be accompanied by such an **F**.) Let the origin of the x, y-coordinate system referred to in Section 1.5.2 lie in the extreme lower left corner of the rectangle covering the primary, i.e., at the point that corresponds to the one designated by x_1, y_1 in the secondary. Thus the secondary is translated by x_1, y_1 against the primary.

Since the secondary is otherwise identical with the primary (except for the addition of **F** in the second case), it will again reproduce (and produce another **F** in the second case), constructing a ternary. This

[30] [There is an interesting parallel between Gödel's undecidable formula (see p. 55 above), which refers to itself, and von Neumann's self-reproducing automaton, which contains a description of itself. See Burks, "Computation, Behavior, and Structure in Fixed and Growing Automata," pp. 19–21.]

will then produce a quaternary, followed by a quinary (each with its concomitant **F** in the second case; cf. above), etc. The shifts involved will be $2x_1$, $2y_1$, then $3x_1$, $3y_1$, then $4x_1$, $4y_1$, etc.

Thus the shift between the p-ary and the q-ary is $(q - p)x_1$, $(q - p)y_1$. Since $p, q = 1, 2, \cdots$, therefore $p \neq q$ implies

$$| q - p | = 1, 2, \cdots,$$

and hence $|(q - p)y_1| = |q - p| \cdot |y_1| \geqq \bar{y}$. Hence, in view of our above observation relating to Section 1.5.2, these two will not intersect. That is, all the successively constructed descendants of the primary will be distinct and non-interfering entities in space. (To be more precise, these descendants will be distinct and non-interfering entities in the underlying crystal [crystalline structure].)

Actually, this program of mutual avoidance among the primary and its descendants must be extended to the paths within the crystal through which each one of these entities, acting as a primary for its own reproduction, connects with the site and operates on the construction of its immediate successor in the line of descent, i.e., its secondary.

This, however, presents no difficulties, and will be gone into in the course of the detailed discussion.

1.7.2.1 Constructed automata: initial state and starting stimulus. The next point to be gone into is that of initiation and timing.

Consider the state of the secondary automaton which the construction is designed to achieve, i.e., its so-called *initial state* {cf. Sec. 1.5.2 immediately after formula (*)}. In all states that lead up to this, and therefore conveniently in this state too, the automaton must be quasi-quiescent. This is clearly necessary for an orderly process of construction, since the already-constructed parts of the not yet completed secondary must not be reactive and changing, while the construction—in adjacent as well as in other areas—is still in progress.

The problem that is encountered here is not unlike the one discussed in Section 1.6.2.3 relative to the quasi-quiescence of **L**. However, it is less severe here. The stimuli that have to be used in exploring **L** must be able to induce responses in the primary, if they are to perform their function of inducing there appropriate actions that depend on the information acquired in inspecting **L**. (This is quite essential to the proper functioning of the primary, as was seen in the discussion of the operation of constructing under the control of instructions in Sec. 1.4.2.1, and again in the discussion of the operation of copying in Sec. 1.6.2.3 and in Sec. 1.6.3.) On the other

hand, the stimuli that create the desired cell states during the construction of the secondary need not have such effects on the class of automata that is involved here, secondary or primary. This will be verified in detail later. Thus it was necessary to keep the "descriptions" **L** of automata sharply apart from the "originals" (cf. Sec. 1.6.3.1), while it will be possible to construct the automata which are relevant here so that they have quasi-quiescent initial states. For the details and precise definitions, cf. later.

The crux of this matter is, of course, that once such a (secondary) automaton is completed, and hence present in its quasi-quiescent initial state, it can then be transferred by some appropriate process of stimulation into its *normal* (i.e., intended) mode of activity. This process of stimulation is most conveniently thought of as a single stimulus, delivered to the appropriate point of the secondary, at the appropriate time after completion, by the primary. This is the secondary's *starting stimulus*. This is, then, the concluding step of the primary in its construction of the secondary. For the details, cf. later.

In the case of self-reproduction, i.e., for the **E** or $\mathbf{E_F}$ discussed in Sections 1.6.1.2 and 1.7.1, the secondary (or one of the secondaries) is a shifted copy of the primary. The starting stimulus activates this secondary and makes it self-reproducing (as the primary had been originally). This, then, maintains the iterative process of self-reproduction with which Section 1.7.1 dealt.

1.7.2.2 Single-action vs. sequential self-reproduction. For the self-reproducing primary (**E** or $\mathbf{E_F}$; cf. above) the next question is this: What does the primary do after it has completed the secondary and given it the starting stimulus?

The simplest arrangement would be to let it return to a quasi-quiescent state which is identical with its original state. Implicitly this is the assumption which fits discussions like the one of continued reproduction in Section 1.7.1.

An alternative possibility is to finish with the quasi-quiescent state, plus activity in a suitable terminal organ which imparts the starting stimulus again. This scheme leads to repeated self-reproductions by the original primary, and of course similarly by all its descendants in the first, second, third, etc. degree. However, it is not operable without one more elaboration.

Indeed, as it stands it would cause the primary to try to form all successive secondaries with the same x_1, y_1, i.e., at the same location in the crystal. This is obviously conflicting; at best the second construction of a secondary would override and destroy the first speci-

men. However, it is even more likely that the first secondary, which by then is reactive, will interfere with the second (attempted) construction, causing an unforeseeable class of malfunctions and corrupting all reproduction. It is therefore necessary to change x_1 , y_1 between the first and the second attempt to construct a secondary, and similarly between the second and the third one, the third and the fourth one, etc., etc. This changing of x_1 , y_1 must therefore take place during (i.e., as a part of) the activity of the terminal organ referred to above. The arithmetical rules that control these successive modifications of x_1 , y_1 must be such that the whole sequence of secondaries of the original primary do not conflict with each other, nor with the possible **F** which accompany them (cf. the first part of Sec. 1.7.1), nor with the paths which are required for their construction (cf. the end of Sec. 1.7.1). In addition, every secondary of the original primary, since it is a shifted copy of the latter, will behave in the same way. Thus a double sequence of ternaries will be constructed from these, then by the same mechanisms a triple sequence of quaternaries, then a quadruple sequence of quinaries, etc., etc. The rules for the successive modifications of the x_1 , y_1 must hence be such that no two in all the orders of this hierarchy ever interfere with each other, or with each other's possible **F**, or with each other's construction paths.

This requirement sounds complicated, but it is not particularly difficult to implement by suitable arithmetical rules concerning the formation of the successive x_1 , y_1 . This will be discussed later.

The above discussion thus distinguishes between two types of self-reproduction: first, when each primary constructs only one secondary, and second, when each primary keeps constructing secondaries sequentially without ever stopping. We will designate these as the *single-action* and the *sequential* type of self-reproduction, respectively.

1.7.3 Construction, position, conflict. Some remarks about physiological analogs of the above constructions are now in order.

Comparing these processes of construction and reproduction of automata, and those of actual growth and reproduction in nature, this difference is conspicuous; in our case the site plays a more critical role than it does in reality. The reason is that by passing from continuous, Euclidean space to a discrete crystal, we have purposely bypassed as much as possible of kinematics. Hence the moving around of a structure which remains congruent to itself, but changes its position with respect to the crystal lattice, is no longer the simple and elementary operation it is in nature. In our case, it would be about as complex as genuine reproduction. This means that all

of our structures are rather rigidly tied to their original location, and all conflicts and collisions between them are primarily conflicts in location.

It is true in the natural setting, too, that conflicts and collisions are due in the same way to location, but there the scheme has more elasticity because of the possibility of motion. The limitations of the pattern due to this circumstance are obviously the price one has to pay for the simplicity obtained by our elimination of kinematics (cf. the discussions of Secs. 1.3.1.1–1.3.3.1).

An essential preliminary condition for the mechanisms of reproduction that we have considered is the quiescence of the area in which they are to function (cf., e.g., the remarks in the first part of Sec. 1.7.2.1 and in the first part of Sec. 1.7.2.2). That is, the region of the crystal surrounding the primary must be free of all reactive organisms, and this must be true as far as the process of reproduction is expected to progress unhindered. It is quite clear that where the reproductive expansion of the area under the influence of the primary collides with other reactive organisms, the "unforeseeable malfunctions" referred to in Section 1.7.2.2 can set in. This is, of course, just another way to refer to the conflict situations involving several independent organisms that have come into contact and interaction.

1.7.4.1 $\mathbf{E_F}$ *and the gene-function.* Another physiological analog worth pointing out is the similarity of the behavior of automata of the $\mathbf{E_F}$ type with the typical gene function.[31] Indeed, $\mathbf{E_F}$ reproduces itself and also produces a prescribed **F**. The gene reproduces itself and also produces—or stimulates the production of—certain specific enzymes.

1.7.4.2 $\mathbf{E_F}$ *and the mutation—types of mutation.* A further property of $\mathbf{E_F}$ that may be commented on is this. Assume that a cell of $\mathbf{E_F}$ is arbitrarily changed. If this cell lies in the **D**-region of $\mathbf{E_F}$, it may inhibit or completely misdirect the process of reproduction. If, on the other hand, it lies in the $\mathbf{L_{D+F}}$ region of $\mathbf{E_F}$, then $\mathbf{E_F}$ will construct a secondary, but this may not be related to it (and to **F**) in the desired manner. If, finally, the altered cell lies in the $\mathbf{L_{D+F}}$ region, and more particularly in the description of **F** within it, modifying **F** into, say $\mathbf{F'}$, then the production of $\mathbf{E_{F'}}$ will take place, and in addition to it an $\mathbf{F'}$ will be produced.

Such a change of a cell within $\mathbf{E_F}$ is rather reminiscent of a *mutation* in nature. The first case would seem to have the essential traits of a lethal or sterilizing mutation. The second corresponds to one without

[31] Von Neumann, "The General and Logical Theory of Automata." *Collected Works* 5.317–318.

these traits, but producing an essentially modified, presumably sterile, successor. The third one produces a successor which is viable and self-reproducing like the original but has a different by-product ($\mathbf{F}'$ instead of $\mathbf{F}$). This means a change of the hereditary strain.

Thus the main classification of mutations turns out to be quite close to the one occurring in nature.

1.8 Evolution—Question (E)

The observations of Section 1.7 tend towards the transition from question (D) to question (E). On (E), itself, the question of evolution, we will only make a few remarks at this point.

There is no difficulty in incorporating logical devices into automata of the types $\mathbf{E}$ or $\mathbf{E}_{\mathbf{F}}$ which will modify the $\mathbf{D}$, $\mathbf{F}$ areas in their $\mathbf{L}_{\mathbf{D}}$, $\mathbf{L}_{\mathbf{D+F}}$, respectively, depending on outside stimuli which they may have received previously. This would amount to a modification of the mass of heredity that they represent by the occurrences (experiences) of their active existence. It is clear that this is a step in the right direction, but it is also clear that it requires very considerable additional analyses and elaborations to become really relevant. We will make a few remarks on this subject later.

In addition to this it must be remembered that conflicts between independent organisms lead to consequences which, according to the theory of "natural selection," are believed to furnish an important mechanism of evolution. As was seen at the end of Section 1.7.3, our models lead to such conflict situations. Hence this motive for evolution might also be considered within the framework of these models. The conditions under which it can be effective here may be quite complicated ones, but they deserve study.

A SYSTEM OF 29 STATES WITH A GENERAL TRANSITION RULE

2.1 Introduction

2.1.1 The model: states and the transition rule. In this chapter we will develop the first model that possesses the potentialities of logical and constructive universality and of self-reproduction (cf. questions (A)–(E) in Sec. 1.1.2.1), as well as the other attributes evolved in the course of the discussion of these in Chapter 1. This model is based on a crystalline medium (cf. Secs. 1.3.3.1–1.3.3.3); we will be able to construct it in two dimensions and to use there the quadratic[1] (regular) lattice {cf. the end of Sec. 1.3.3.3, in particular questions (P) and (R)}. Each lattice point of this crystal will be able to assume a finite number of different states (say N states) and its behavior will be described (or controlled) by an unambiguous *transition rule*, covering all transitions between these states, as affected by the states of the immediate neighbors.

We will, then, perform the major constructions called for by questions (A)–(E) in Section 1.1.2.1 (and the relevant subsequent discussions of Ch. 1) for a specific model defined along these lines.

2.1.2 Formalization of the spatial and the temporal relations. At this point we introduce some rigorous concepts and notations.

The lattice points of the quadratic crystal (cf. Sec. 2.1.1) are designated by two integer-valued coordinates, i, j. It is natural to treat the crystal as unlimited in all directions, at least as long as there does not emerge some definite reason for proceeding differently. This determines the ranges of i, j:

$$i, j = 0, \pm 1, \pm 2, \cdots . \tag{1}$$

[It does not matter which lattice point is selected as the origin $(0, 0)$.] The pair i, j thus represents a point in the plane, but it is also convenient to view it as a vector, i.e., to treat it as an additive quan-

[1] [The lattice points of a quadratic crystal lie at the corners of squares.]

tity. We write

$$\vartheta = (i, j). \tag{2}$$

The nearest neighbors of (i, j) are the four points $(i \pm 1, j)$, $(i, j \pm 1)$. The next neighbors are the four points $(i \pm 1, j \pm 1)$. In Figures 4a and 4c the nearest neighbors of $\times$ are marked with small circles ($\circ$), and the next nearest neighbors are marked with heavy dots ($\bullet$).

Put

$$\begin{cases} v^0 = (1, 0), & v^1 = (0, 1), \\ v^2 = -v^0 = (-1, 0), & v^3 = -v^1 = (0, -1), \end{cases} \tag{3}$$

and

$$\begin{cases} v^4 = (1, 1), & v^5 = (-1, 1), \\ v^6 = -v^4 = (-1, -1), & v^7 = -v^5 = (1, -1). \end{cases} \tag{4}$$

See Figure 4b. The nearest neighbors of ϑ are the $\vartheta + v^\alpha$ ($\alpha = 0, \cdots, 3$), and the next neighbors of ϑ are the $\vartheta + v^\alpha$ ($\alpha = 4, \cdots, 7$). One might hesitate as to whether the immediate neighbors of ϑ referred to in Section 2.1.1 should be, among the $\vartheta + v^\alpha$, the four with $\alpha = 0, \cdots, 3$ or the eight with $\alpha = 0, \cdots, 7$. We will choose the former, since it leads to a simpler set of tools.

In Figures 4a and 4b the crystal lattice was shown in the usual manner, the lattice points being the intersections of the lines. In future figures we will use a different scheme: the lattice points will be shown as squares, and *immediate neighbors* (which in Figs. 4a and 4b were connected by single edges) will now be squares in contact (i.e., with a common edge). Furthermore, we will always show only those squares (i.e., only those lattice points) which are needed to illustrate the point immediately at hand. Thus Figure 4a assumes the appearance of Figure 4c.

As discussed in Section 1.2.1, the range of time t is

$$t = 0, \pm 1, \pm 2, \cdots. \tag{5}$$

Each lattice point is a cell in the sense of Sections 1.3.3.1 and 1.4.1.1. It is able to assume N states (cf. Sec. 2.1.1); let these be designated by an index

$$n = 0, 1, \cdots, N - 1. \tag{6}$$

The state of cell $\vartheta = (i, j)$ at time t will therefore be written

$$n_\vartheta{}^t. \tag{7}$$

Also, the N numerals $0, 1, \cdots, N-1$ used in expression (6) to designate n may be replaced by any other N symbols, according to convenience.

The system is to be intrinsically homogeneous in the sense of Section 1.3.3.2; i.e., the same rule will govern its behavior at each lattice point ϑ. This rule is the *transition rule* referred to in Section 2.1.1, which defines the state of the cell ϑ at time t in terms of its own state and of the states of its immediate neighbors at suitable previous times. We will limit and simplify our system by restricting these "suitable previous times" to precisely the immediate predecessor of t, i.e., $t-1$. Thus $n_\vartheta{}^t$ will be a function of $n_\vartheta{}^{t-1}$ and of the $n_{\vartheta+v^\alpha}^{t-1}$ $(\alpha = 0, \cdots, 3)$. That is,

$$n_\vartheta{}^t = \mathbf{F}(n_\vartheta{}^{t-1}; n_{\vartheta+v^\alpha}^{t-1} \mid \alpha = 0, \cdots, 3). \tag{8}$$

Let m take the place of $n_\vartheta{}^t$ and let m^α take the place of $n_{\vartheta+v^\alpha}^t$. The function $\mathbf{F}$ then becomes $\mathbf{F}(m; m^\alpha \mid \alpha = 0, 1, 2, 3)$. This N-valued function $\mathbf{F}$ of five N-valued variables represents, therefore, the transition rule. It is the sole and complete rule that governs the behavior of this (intrinsically homogeneous) system.

Note that the range of $\mathbf{F}$ has N elements, while the domain of $\mathbf{F}$ (the set of all quintuplets) has N^5 elements. Hence there are

$$N^{(N^5)} \tag{9}$$

possible functions $\mathbf{F}$, i.e., this many possible transition rules, or models of the class under consideration.

2.1.3 Need for a pre-formalistic discussion of the states. Let us now discuss in a more heuristic way what the N states of a cell should be. The nature of these states is, of course, described not by their enumeration (6), but by the transition rule (8). The only relevant information contained in (6) is the number of states, N. In accord with this, the rigorous summation of these considerations will consist of a specification of the transition rule (8), i.e., of the function $\mathbf{F}$. In the present, heuristic stage, however, it will be better to proceed with an enumeration (6), attaching to each n of (6) a name and a verbal description of the role that it is intended to play. In this connection we will also make use of the possibility of notational changes, referred to in the remark after (6) and (7) in Section 2.1.2.

2.2 Logical Functions—Ordinary Transmission States

2.2.1 Logical-neuronal functions. To begin with, states are needed to express the properly logical or neuronal functions, as discussed in Section 1.2.1. This calls for the equivalents of the neurons of Figure 3 and of their connecting lines.

2.2.2.1 Transmission states—connecting lines. We consider first the connecting lines. These must now be rows of cells, i.e., of lattice points. Since a line must be able to pass a (neural) stimulus, each one of its cells must possess, for this purpose alone, a *quiescent* and an *excited* state. The purpose that we consider here is to transmit a (neural) stimulus. We call these therefore the *transmission states* of the cell and designate them by the symbol **T**. We use an index $\epsilon = 0, 1$; i.e., we write $\mathbf{T}_\epsilon$ to indicate quiescence and excitation. Let $\epsilon = 0$ designate the former and $\epsilon = 1$ the latter.

This transmission must be a directed process, since the lines (that the cells in transmission states replace) were directed to connect definite points. Hence we must set up certain limitations. We may stipulate that a cell in a transmission state accepts a stimulus only from one, definite direction, its *input direction*. That is, an excited transmission cell brings an immediate neighbor (which is a quiescent transmission cell) into the excited transmission state (or, if the latter is found in that state, it keeps it there) only if the former lies in the latter's input direction. Alternatively, we may also stipulate that a cell in a transmission state emits a stimulus only in one, definite direction, its *output direction*. That is, an excited transmission cell brings an immediate neighbor (which is a quiescent transmission cell) into the excited transmission state (or, if the latter is found in that state, it keeps it there) only if the latter lies in the former's output direction. Finally, we may make both stipulations together.

After trying various models along these lines, it appeared most convenient to stipulate a definite output direction. In order to avoid certain uncontrolled, and hence undesirable, return-stimulation phenomena, it seems desirable, while not prescribing any particular input direction, to specify that the output direction is insensitive to inputs.

The $v^\alpha (\alpha = 0, \cdots, 3)$ of Figure 4b enumerate all possible directions for an immediate neighbor (cf. the remarks after expressions (3) and (4) in Sec. 2.1.2). Hence the $\mathbf{T}_\epsilon$ will be given a further index $\alpha = 0, \cdots, 3$: $\mathbf{T}_{\alpha\epsilon}$, so that $\mathbf{T}_{\alpha\epsilon}$ has the output direction v^α. The above stipulations now assume this form: $\mathbf{T}_{\alpha' 1}$ at ϑ' induces $\mathbf{T}_{\alpha 1}$ at ϑ (from $\mathbf{T}_{\alpha 0}$ or $\mathbf{T}_{\alpha 1}$) if and only if $\vartheta = \vartheta' + v^{\alpha'}$, but $\vartheta' \neq \vartheta + v^\alpha$, i.e., if and only if $\vartheta - \vartheta' = v^{\alpha'} \neq -v^\alpha$.

Let us now use the symbols $\mathbf{T}_{\alpha\epsilon}$ $(\alpha = 0, \cdots, 3; \epsilon = 0, 1)$ in place of some eight number values in expression (6) (cf. the remark after expression (6) and (7) in Sec. 2.1.2). Let us also recognize the unit time delay of the stimulus-response process, as discussed in Section 2.1.2. Then the above rule becomes:

$$(10)\qquad \begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{T}_{\alpha\epsilon}\,. \\ \text{Then } n_\vartheta{}^{t} = \mathbf{T}_{\alpha 1} \text{ if } n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha' 1} \\ \text{for some } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'} \neq -v^{\alpha}. \\ \text{Otherwise } n_\vartheta{}^{t} = \mathbf{T}_{\alpha 0}\,. \end{cases}$$

2.2.2.2 Delays, corners, and turns in connecting lines. Note that this model for the connecting lines differs from the one considered in Section 1.2.1 in that it introduces finite propagation delays. We have now a unit time delay between immediate neighbors. However, this deviation from the pattern of Section 1.2.1 will have no relevant undesirable consequences.

Note also that this model serves equally to synthesize straight connecting lines from transmission cells, and connecting lines with corners or turns in them. Straight lines are shown in Figures 5a–5d; these represent the four possible "straight" directions in our lattice. "Corners" and "turns" are shown in Figures 5e and 5f. Figures 5a–5f are drawn according to the rules stated in Figure 4c. Figures 5a′–5f′ are simplified (and more easily readable) versions of Figures 5a–5f, respectively, in which each $\mathbf{T}_{\alpha\epsilon}$ is replaced by the arrow of its v^{α} (cf. Fig. 4b).

We consider next the specific neurons of Figure 3.

2.3 Neurons—Confluent States

2.3.1 The + neuron. The + neuron merely calls for an excitable cell which has an output and two possible inputs. There is, of course, no harm done if it can accommodate more than two such inputs. We defined a transmission cell in Section 2.2.2.1 so that it has three possible inputs. Every one of its four sides, excepting the output side, is an input. Thus our transmission cells not only fulfill the function of (elements of) connecting lines between neurons (this being the function for which they were originally intended), but also fill the role of + neurons.

The use of an ordinary transmission cell as a + neuron, i.e., as a connecting line junction, is shown in Figure 5g. This figure is drawn according to the scheme of Figures 5a′–5f′.

2.3.2 Confluent states: the · neuron. The · neuron calls for an excitable cell that has an output and two inputs, which must be stimulated together in order to produce excitation. It would be quite practical to introduce a class of such states. However, a free choice of an output direction and two input directions (from the totality of four possible directions, as represented by the v^{α}, $\alpha = 0, \cdots, 3$), would require $(4 \times 3 \times 2)/2 = 12$ kinds, and, since there must be a quiescent and an excited state for each kind, a total of 24 states. It is possible to

achieve equally satisfactory results with more economy, namely with only one kind, and hence with two states. This can be done by prescribing no particular input or output directions at all, i.e., by stipulating that every direction is a possible input, as well as a possible output. In addition to this, one can then prescribe as the prerequisite of excitation a minimum of two stimulations, i.e., a minimum of two excited transmission cells that are immediate neighbors and in whose output direction our cell lies. However, it is still more convenient to frame this condition more elastically and to stipulate that the cell under consideration gets excited if every immediately neighboring transmission cell, whose output direction points at this cell, is itself excited. (This is to be taken with the exclusion of the subcase—which is strictly logically admissible, but obviously conflicting with the intention—that none of the immediate neighbors qualifies, i.e., is a transmission cell and has its output direction pointing at this cell.) This formulation of the rule has the effect that the cell under consideration can act as a neuron of threshold one (i.e., from the point of view of inputs, like an ordinary transmission cell), or two (i.e., like the desired · neuron), or three (i.e., like a combination of two · neurons), depending on whether one, two, or three, respectively, of its immediate neighbors are transmission cells with their output directions pointing at it. (Since this cell should not be able to stimulate any transmission cell whose output direction is pointing at it—cf. rule (10) and its adaptation to the present situation in rule (12) below—it would be pointless to have all of its four immediate neighbors in such a state. This situation would preclude any results of an excitation of our cell.)

We will call these states of a cell *confluent states*, and designate them by the symbol **C**. We again use the index $\epsilon = 0, 1$; that is, we write $\mathbf{C}_\epsilon$ to indicate quiescence ($\epsilon = 0$) and excitation ($\epsilon = 1$). We proceed now similarly as we did at the end of Section 2.2.2.1. We use the symbols $\mathbf{C}_\epsilon$ ($\epsilon = 0, 1$) in place of two number values in expression (6) (cf. the remark after expressions (6) and (7) in Sec. 2.1.2). The rule that we formulated above now becomes, inasmuch as it affects the inputs to **C**:

(11)
$$\begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{C}_\epsilon. \\ \text{Then } n_\vartheta{}^{t} = \mathbf{C}_1, \text{ if both (a), (b) hold:} \\ \quad \text{(a) } n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha' 1} \text{ for some } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'}. \\ \quad \text{(b) Never } n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha' 0} \text{ for an } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'}. \\ \text{Otherwise } n_\vartheta{}^{t} = \mathbf{C}_0. \end{cases}$$

The portion of the rule that affects the outputs of **C** must be stated

as a modification of rule (10), since it provides for a new way to excite a transmission cell, i.e., to produce a $\mathbf{T}_{\alpha 1}$ from a $\mathbf{T}_{\alpha\epsilon}$. This is expressed by the following insertion between the second and third sentences of rule (10):

$$(12) \qquad \begin{cases} \text{Also } n_{\vartheta}{}^{t} = \mathbf{T}_{\alpha 1} \quad \text{if} \quad n_{\vartheta'}{}^{t-1} = \mathbf{C}_1 \text{ for some } \vartheta' \\ \text{with } \vartheta - \vartheta' = v^{\beta} \neq -v^{\alpha} \qquad (\beta = 0, \cdots, 3). \end{cases}$$

Note that the system of rules (10), (11), and (12) provides for excitation of **T** by **T**, also of **C** by **T**, and **T** by **C**, but not for an excitation of **C** by **C**. This arrangement will have no relevant undesirable consequences.

A · neuron with its close surroundings is shown in Figure 6a. This figure is drawn according to the scheme of Figures 5a′–5f′ and 5g. The confluent state **C** here makes its first appearance.

2.3.3 The − neuron. The − neuron calls for an excitable cell in which the roles of quiescence and excitation are interchanged in comparison with the transmission states. It must be ordinarily excited (i.e., able to excite an immediately neighboring cell in a transmission state, at which its output direction points), but it must be made quiescent by an input stimulation (reverting to excitation when the stimulation ceases). We could introduce a class of such states—e.g., with a given output direction, all other directions being input directions, just as in the transmission states. Since there are four possible directions, this would require four kinds, and with quiescence and excitation for each kind, eight states would be needed. However, we are reluctant to introduce a class of such states whose ordinary, unperturbed condition is not quiescence. This objection could be circumvented in various ways, with higher or lower degrees of economy.[2] We shall find that a class of states which we will introduce later for other reasons can be used to synthesize the function of the − neuron. We can therefore forego altogether taking care of it at this stage.

2.3.4 The split. Although all the neuron species of Figure 3, as

[2] [Von Neumann here referred to the double line trick of his "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Collected Works* 5.337. Using only + neurons and · neurons, he synthesized a complete set of truth-functional primitives by using a pair of lines with the codings 01 (for "zero") and 10 (for "one"). In other words, each line of the pair is in the opposite state from the other line of the pair, so that negation may be realized by interchanging (crossing) the two lines of a pair. But in the present manuscript von Neumann synthesized negation from the destructive (reverse) and constructive (direct) processes of Secs. 2.5 and 2.6 below. An example of this synthesis is given in Fig. 17 of Sec. 3.2 below.]

well as their connecting lines, have been disposed of, there remains one more entity in this category to consider. Logical (i.e., neuronal) networks in the sense of Section 1.2.1 must in most cases contain connecting lines that lead from one output to several inputs, i.e., output lines that have to be split. (This was mentioned in Sec. 1.2.1.)[3] That is, transmission-like states with several outputs are needed.

However, it suffices to note that our definition of the confluent states takes care of this need. (Cf. Sec. 2.3.2, in particular the discussion of the "threshold 1" geometry for this class of states, near the end of Sec. 2.3.2. In this condition the cell has one input, and therefore up to three possible outputs.)

A split that is achieved by using the confluent states is shown in Figure 6b. This figure is drawn according to the scheme of Figure 6a. This is also true of all subsequent figures, with such exceptions and modifications as are expressly stated.

2.4 Growth Functions: Unexcitable State and Special Transmission States

2.4.1 Muscular or growth functions—ordinary vs. special stimuli. Having finished with the logical (i.e., neuronal) functions in the sense of Section 1.2.1, we can now pass to the others. In Section 1.2.2 these were temporarily designated as muscular, but the discussion of Section 1.3 showed that they are more properly viewed and treated as growth functions (cf. in particular Secs. 1.3.4.2 and 1.3.4.3). At any rate, we need states to express these functions.

We know (cf. the above references) that this leads over into the problem of ordinary vs. special stimuli, i.e., of ordinary vs. special excited states. The ordinary class is the one used for logical purposes, i.e., the one considered up to now (specifically in Secs. 2.1.3–2.3.2). The special class is the one that we will have to introduce now, in order to take care of the functions referred to above.

2.4.2 Unexcitable state. The purpose of the special class of excited states is to induce growth in the sense of Sections 1.3.4.2 and 1.3.4.3, i.e., to transfer cells from unexcitable to excitable states, and within the latter category also to determine the state's species. As for the last mentioned determination, we have already stipulated the existence of several species: the transmission, or **T** states, which formed four species, the $\mathbf{T}_\alpha$, $\alpha = 0, 1, \cdots, 3$; and the confluent species **C** (cf. Secs. 2.2.2.1 and 2.3.2). Note that each of these actually corre-

[3] [In his list of footnotes von Neumann wrote here "Degeneration (?)." I do not know what he intended. Possibly he was going to say that power amplification is needed when one input line drives two or more output lines.]

sponds to two states, with $\epsilon = 0, 1$ (quiescent and excited): $\mathbf{T}_{\alpha\epsilon}$, $\mathbf{C}_\epsilon$. We need nevertheless refer to the species $\mathbf{T}_\alpha$, $\mathbf{C}$ only, or, to be more precise, to the states with $\epsilon = 0$ only: $\mathbf{T}_{\alpha 0}$, $\mathbf{C}_0$. The reason is that it suffices to be able to create each excitable species in its quiescent state. If the excited state is wanted, it can be induced later by ordinary stimuli (for the latter, cf. Secs. 1.3.4.2 and 2.4.1).

It is therefore necessary to introduce the unexcitable state (unexcitable by ordinary stimuli, cf. the above observations and references). We will designate it by **U**. Before we give a rigorous account of its properties, however, we must discuss some connected matters.

2.4.3 Direct and reverse processes—special transmission states. It is desirable not only to be able to effect transfers from the unexcitable state **U** into excitable states (for example, **T** and **C**; cf. the discussion in Sec. 2.4.2), but also to have this process reversible, that is, to be able to effect transfers from the excitable states into **U**. Various uses of this two-way operability will appear later. Now the stimuli which induce all these metamorphoses must be transmitted by cell states which are not themselves affected by them in such a way. It is therefore advisable to introduce a new class of transmission states, say $\mathbf{T}'$, which are in their relationship to each other analogs of the **T**. We will therefore have eight such states: $\mathbf{T}'_{\alpha\epsilon}$ ($\alpha = 0, \cdots, 3$; $\epsilon = 0, 1$)—in analogy to the $\mathbf{T}_{\alpha\epsilon}$ of Section 2.2.2.1. Accordingly, we proceed the way we did for the $\mathbf{T}_{\alpha\epsilon}$, i.e., in rule (10) of Section 2.2.2.1.

We use the symbols $\mathbf{T}'_{\alpha\epsilon}$ ($\alpha = 0, \cdots, 3$; $\epsilon = 0, 1$) in place of some eight-number values in expression (6) (cf. the remark after expressions (6) and (7) in Sec. 2.1.2). The rule is:

(13)
$$\begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{T}'_{\alpha\epsilon}. \\ \text{Then } n_\vartheta{}^{t} = \mathbf{T}'_{\alpha 1} \quad \text{if} \quad n_{\vartheta'}{}^{t-1} = \mathbf{T}'_{\alpha' 1} \text{ for some } \vartheta' \\ \text{with } \vartheta - \vartheta' = v^{\alpha'} \neq -v^{\alpha}. \\ \text{Otherwise } n_\vartheta{}^{t} = \mathbf{T}'_{\alpha 0}. \end{cases}$$

2.5 The Reverse Process

2.5.1.1 The reverse process for the ordinary states. We can now give a rigorous definition of the reverse process, that is, the transfer of an excitable state (**T**, **C**) into an unexcitable one (**U**) by the special stimuli (of $\mathbf{T}'$). This takes the form of modifications of the **T** rules (10), (12) and of the **C** rule (11):

(14)
$$\begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{T}_{\alpha\epsilon} \quad \text{or} \quad \mathbf{C}_\epsilon. \\ \text{Then the following rule overrides the rules of (10), (12) and} \\ \quad \text{of (11):} \\ n_\vartheta{}^{t} = \mathbf{U} \quad \text{if} \quad n_{\vartheta'}{}^{t-1} = \mathbf{T}'_{\alpha' 1} \quad \text{for some } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'}. \end{cases}$$

Note that $\vartheta - \vartheta' \not\equiv -v^{\alpha}$ {in the **T** case, cf. the similar stipulations in rules (10) and (12)} is not required; that is, the "kill" by a special stimulus (from a $\mathbf{T}'$) is efficacious even in the output direction (of **T**).

2.5.1.2 The reverse process for the special states. The reasons which make the existence of the reverse process (from excitable to unexcitable; cf. Sec. 2.4.3 and later) desirable for the ordinary excitable states (**T**, **C**), operate in the case of the special ones ($\mathbf{T}'$) too. However, rule (14) could not be extended to the $\mathbf{T}'$ states; such an effect of a $\mathbf{T}'$ on a $\mathbf{T}'$ is inadmissible for the same reason for which the corresponding effect of a **T** on a **T** is inadmissible (cf. Sec. 2.4.3): it would destroy the character of the $\mathbf{T}'$ as transmission states for special stimuli, just as the corresponding effect would have destroyed the character of the **T** as transmission states for ordinary stimuli. The latter circumstance caused us to introduce the $\mathbf{T}'$ (to transfer the **T**, as well as the **C**, into **U**); we can now make a similar use of the **T** (to transfer the $\mathbf{T}'$ into **U**). It is advisable, however, not to endow the **C**, too, with this property. Indeed, since every direction is an output direction for **C**, giving this faculty to **C** would make it much more difficult to direct and to control than is desirable. We introduce therefore this rule, which modifies (13):

(15) Assume $n_{\vartheta}{}^{t-1} = \mathbf{T}'_{\alpha\epsilon}$.
Then the following rule overrides the rules of (13):
$n_{\vartheta}{}^{t} = \mathbf{U}$ if $n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha'1}$ for some ϑ' with $\vartheta - \vartheta' = v^{\alpha'}$.

The remark at the end of Section 2.5.1.1, concerning the outputs, applies here, too.

2.5.2 Origination of special stimuli. We have not provided so far for the origination of special pulses, i.e., for the excitation of the $\mathbf{T}'$, except by each other.

It is not necessary to introduce a complete (logical) neuronic system for special stimuli, as we did for ordinary stimuli in Sections 2.2.2.1–2.3.4. We can handle all of logics with ordinary stimuli, as was indeed intended in introducing them (cf. Secs. 1.3.4.2 and 1.3.4.3), and use these to start chains of special stimuli, whenever necessary. (The physiological analog of this is that logics are restricted to neuronal activity, and that muscular activity is always induced and controlled by neuronal activity. See Sec. 1.2.2 for the simile.) Hence we need a class of states which can respond to an ordinary stimulus by emitting a special one, i.e., which can be excited by a **T** and can excite a $\mathbf{T}'$.

Before introducing a new class for this purpose, let us see whether we might not do it with the existing ones. **T** does not excite $\mathbf{T}'$ {it

"kills" it; cf. rule (15)}; hence we can use neither **T** nor **T**′ for this purpose. This leaves only **C**. Now **C** is excited by a **T**; hence we need only endow it with the ability to excite a **T**′. This can be done. (That **T**′ does not excite a **C**—but "kills" it; cf. Sec. 2.5.1.1—is irrelevant in this connection.) All we need is to stipulate the analog of rule (12) for **T**′. Thus the duality of **T** and **T**′, already expressed by the duality of rules (10) and (13), and by the duality of rules (14) and (15), is further completed by the duality of (12) and of (16) that we will now state. This is to be viewed as an insertion between the second and third sentences of (13), and it is therefore, like the original (13), overridden by (15). The rule follows:

$$(16)\quad \begin{cases} \text{Also } n_\vartheta{}^t = \mathbf{T}'_{\alpha 1}\text{, if } n_{\vartheta'}{}^{t-1} = \mathbf{C}_1\text{, for} \\ \text{some } \vartheta' \text{ with } \vartheta - \vartheta' = v^\beta \neq -v^\alpha \end{cases} \qquad (\beta = 0, \cdots, 3).$$

2.6 The Direct Process—Sensitized States

2.6.1 The direct process. The reverse process of Section 2.4.3 (transfer from excitable to unexcitable) having been taken care of, we pass now to considering the (primarily required) direct process of Section 2.4.2 (transfer from unexcitable to excitable).

The list of species that this process must be able to create has been extended. We had the $\mathbf{T}_\alpha$ and **C** in Section 2.4.2; to these we must add, since Section 2.4.3, the $\mathbf{T}'_\alpha$. In other words, we must be able to create the following states (for the role of $\epsilon = 0$ compare the discussion in Sec. 2.4.2):

$$(17)\qquad \mathbf{T}_{\alpha 0}\text{, } \mathbf{T}'_{\alpha 0}\text{, } \mathbf{C}_0\text{.}$$

This is a total of nine states.

Thus we need a mechanism to transfer **U** into any one of the nine states of (17). Regarding this mechanism two remarks are in order.

2.6.2.1 First remark: duality of ordinary and special states. We have two kinds of stimuli at our disposal: ordinary and special, corresponding to the excitation of the **T** (possibly together with the **C**) and of the **T**′ states, respectively. Our original intention had been to use only the special stimuli, i.e., the **T**′ states, for transforming **U** into any one of the nine states of (17) (cf. Secs. 2.4.1 and 2.4.2). Subsequently however, in dealing with the reverse process (cf. Secs. 2.4.3–2.5.2), we let the **T** (actually without the **C**) and the **T**′ states play rather complementary and symmetric roles with respect to each other (cf. the references to "duality" at the end of Sec. 2.5.2). It is therefore tempting to assign to them similarly symmetric roles in connection with the direct process.

This might again be done in a dual way, i.e., by using the **T**′ for

transfers from **U** to **T** (and to **C**) and the **T** for transfers from **U** to **T**′. However, even this limitation will prove to be unnecessary, and a satisfactory system can be built by stipulating that **T** and **T**′ have identical and interchangable effects on **U** for transfers into all states **T**, **C**, and **T**′ {i.e., all nine states of (17)}.

This raises the question of why **T**′ had to be introduced at all, if **T** alone can induce and control all transfers from **U**. The answer is that the reverse process called for **T**′ because of the contrast between transmission and causing transfer into **U**; compare rules (10) and (12) with rule (14), and rules (13) and (16) with rule (15). Moreover, the reverse process (transfer from **T**, **T**′, or **C** into **U**) is a prerequisite for a proper control of the direct process (transfer from **U** into **T**, **T**′, or **C**—or even into **T** or **C** only). This point deserves somewhat closer consideration.

2.6.2.2 The need for the reverse process. In Figure 7, cells 1, · · · , 9 constitute an area of 3 × 3 cells (i.e., lattice points; cf. the explanation of Fig. 4c) which is to be organized, that is, transferred from the **U** states into various (prescribed) states, e.g., **T** states. This organization of cells 1, · · · , 9 is to originate from and be controlled by area ⊙.

Consider the transfer of the middle cell, number 5. Whether this is done by ordinary or by special pulses (i.e., by **T** or by **T**′ excitations), an unbroken chain of (**T** or **T**′) transmission cells must be laid down, from the area of the origination and logical control of these excitations to the cell to be operated on, in this case number 5. In Figure 7 the cells marked with arrows form the chain; they may be $\mathbf{T}_\alpha$ according to Figure 5 or their corresponding $\mathbf{T}'_\alpha$. This chain must cross the ring of cells that surround the target cell number 5, i.e., the ring of cells numbered 1, · · · , 4, 6, · · · , 9. In Figure 7 the chain crosses at cell number 8.

Now the desired organization of the area 1, · · · , 9 may provide for the cell of the cross-over (in this case number 8; cf. above) another (quiescent) state than that of the chain. (The latter must be a $\mathbf{T}_\alpha$ or a $\mathbf{T}'_\alpha$ state, with v^α in the direction of the chain; this is $\alpha = 1$ in the present case; cf. Figs. 4b and 7.) Hence the organization of this cell cannot have occurred before that of cell number 5. If it is to occur after that of cell number 5, then the (**T** or **T**′) cell of the chain that is involved must be transferred into the desired state. Since the direct process allows the transfer from **U** into any desired state, it is simplest to provide for a way to transfer from the present (**T** or **T**′) state into **U**. Hence the reverse process is indeed necessary.

2.6.3.1 Second remark: the need for fixed stimulus sequences to control the direct process. We observed near the end of Section 2.6.2.1

that the direct process {i.e., the transfer from **U** into the states of list (17)} should be effected by the stimuli due to **T** and **T′** excitations, and that it will prove possible to let both (i.e., **T** and **T′**) have exactly the same effects in this respect. It will also appear later that this arrangement is more economical in some relevant ways than its obvious alternatives.

It is again advisable to exclude the **C** from this process. The reasons for using the direct process here are the same as those given (in Sec. 2.5.1.2) for using the reverse process. Note in addition that, just because every direction is an output direction for **C**, it is often necessary to protect certain sides of a **C**, and for this function a **U** is natural (cf. later). Hence it would be most inconvenient if a **C** had any effect on a **U**.

The direct process must provide for transfers from **U** into every one of the nine states of list (17). These nine alternatives are too many to be handled by a single stimulus, even if **T** and **T′** excitations had different effects. Besides, we stipulated that **T** and **T′** have the same effects in this process. Hence the nine alternatives which are now needed must be expressed by some binary coded stimulus sequence. In this binary code a digit 1 is expressed by a stimulus (**T** or **T′**), while a digit 0 is expressed by the absence of a stimulus. Coded sequences of length three can express eight alternatives: 000, 001, $\cdots$, 111. Since nine alternatives are wanted, one of these—e.g., the first one, 000—must be made to express two further alternatives; i.e., it must be continued to 0000 and 0001. Thus we end up with the nine coded sequences

$$\begin{cases} 0000,\ 0001,\ 001,\ 010,\ 011 \\ 100,\ 101,\ 110,\ 111. \end{cases} \tag{18}$$

This must be made to correspond to the nine states of list (17).

The nine coded sequences of list (18) will have to be built up successively from their constituent digits 0 and 1. That is, in the (direct) process of transfer, **U** will have to go through intermediate states corresponding to the subsequences encountered in building up the nine coded sequences of list (18). These coded subsequences are

$$\begin{cases} 0,\ 1,\ 00,\ 01, \\ 10,\ 11,\ 000; \end{cases} \tag{19}$$

i.e., their number is seven. Finally, there must be a state corresponding to the beginning of the process, where the formation of the coded sequence {going through subsequences of list (19) to a sequence of list (18)} has not yet begun. Here the coded subsequence is best

interpreted as being (in its initial condition) the empty set, to be designated

(20) θ.

2.6.3.2 Additional states required. Let us use a common symbol Σ for the 17 coded sequences of lists (20), (19), and (18). These must correspond to 17 states to be designated $\mathbf{S}_\Sigma$. However, the $\mathbf{S}_\Sigma$ with the 9 Σ of list (18) must be the 9 states of list (17); i.e., these are not new states. In addition, we must consider whether it is not natural and proper to make $\mathbf{S}_\Sigma$ with the Σ of (20) (i.e., $\mathbf{S}_\theta$) coincide with **U**.

The direct process goes from $\mathbf{S}_\theta$ through the $\mathbf{S}_\Sigma$ of list (19) to the $\mathbf{S}_\Sigma$ of list (18) {i.e., to the states of list (17)}, by adding a digit 1 to Σ when a (**T** or **T**′) stimulus occurs, and by adding a digit 0 to Σ when no stimulus occurs. {Of course, this process of increasing Σ ceases to operate when Σ has reached its maximum size (18). Then we deal with the states of list (17), and these are governed by the rules (10)–(16).} This means that the evolution from $\mathbf{S}_\theta$ to the final $\mathbf{S}_\Sigma$ {with Σ maximal, i.e., according to list (18)} is rigidly timed. The absence of the stimulus has as definite an effect as the presence of one, and therefore the stimuli that are required must be delivered at definite times, without any delays other than the expressly required ones. On the other hand, **U** was always conceived of as a quiescent state; it should not change unless it receives a stimulus. Combining these two facts, we see therefore that **U** and $\mathbf{S}_\theta$ must not be identified. Thus we come out with eight new states, namely the $\mathbf{S}_\Sigma$ with the Σ of lists (20) and (19).

2.6.4 Sensitized states. We call the new states, to which Sections 2.6.3.1 and 2.6.3.2 lead us, *sensitized states*. To restate, these are the $\mathbf{S}_\Sigma$, with Σ according to lists (20) and (19), but not according to list (18) {the latter being the old states of list (17)}.

The rigorous rules that control the behavior of **U** and of the sensitized states can now be extracted from the discussions of Sections 2.6.3.1 and 2.6.3.2. These rules are:

(21)
$$\begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{U}. \\ \text{Then } n_\vartheta{}^{t} = \mathbf{S}_\theta \text{ if } n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha'1} \text{ or } \mathbf{T}'_{\alpha'1} \\ \text{for some } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'}. \\ \text{Otherwise } n_\vartheta{}^{t} = \mathbf{U}. \end{cases}$$

(22)
$$\begin{cases} \text{Assume } n_\vartheta{}^{t-1} = \mathbf{S}_\Sigma \text{ with a } \Sigma \text{ of (20) or (19).} \\ \text{Then } n_\vartheta{}^{t} = \mathbf{S}_{\Sigma 1} \text{ if } n_{\vartheta'}{}^{t-1} = \mathbf{T}_{\alpha'1} \text{ or } \mathbf{T}'_{\alpha'1} \\ \text{for some } \vartheta' \text{ with } \vartheta - \vartheta' = v^{\alpha'}. \\ \text{Otherwise } n_\vartheta{}^{t} = \mathbf{S}_{\Sigma 0}. \end{cases}$$

[Note that $\mathbf{S}_{\theta 0} = \mathbf{S}_0$, $\mathbf{S}_{\theta 10} = \mathbf{S}_{10}$, $\mathbf{S}_{\theta 111} = \mathbf{S}_{111}$, etc.]

2.7 Even and Odd Delays

2.7.1 Even delays by path differences. We have completed the description of the direct process of Section 2.4.2 (cf. Sec. 2.6), i.e., of the process that operates the transfers from **U** into the states of list (17). We noted in Section 2.6.3.2 that the direct process is rigidly timed, that is, that each state of (17) calls for certain stimuli delivered at definite distances in time from each other. To be more specific, to get from $\mathbf{S}_\theta$ to the states of list (17), i.e., to the $\mathbf{S}_\Sigma$ with Σ from list (18), the uninterrupted stimulus-no-stimulus sequences of list (18) are needed, according to rule (22). These are certain sequences of lengths three and four. Since we actually want to start from **U**, we must also apply rule (21); that is, we must have one stimulus immediately before this sequence. Hence the complete requirement prescribes certain uninterrupted stimulus-no-stimulus sequences of lengths four and five.

In a properly organized control system such a sequence will be induced by a suitable (single) control stimulus. To get a prescribed train of stimuli during four or five consecutive moments t {t is integer valued; cf. expression (5)} from a (single) control stimulus, some multiple delay system is needed. See Figure 8. If the (single) control stimulus appears at a lattice point **A**, and if the prescribed stimulus-no-stimulus sequence is wanted at a lattice point **B**, then it is necessary to route the excitation from **A** to **B** over several paths, which produce, relatively to each other, definite delay differences. Indeed let $\mathcal{P}$ be the path from **A** to **B**, over which the excitation arrives first to **B**. Then the other paths from **A** to **B**, say $\mathcal{P}_1$, $\mathcal{P}_2$, $\cdots$, must produce delays against $\mathcal{P}$ that are equal to the distances of the stimuli prescribed in the desired stimulus-no-stimulus sequence from its first stimulus (the one required by rule (21); cf. above). Figure 8 exemplifies such a situation. [The delay on the output side of cell **A** to an input side of cell **B** is 7 along path $\mathcal{P}$, 17 along path $\mathcal{P}_1$, and 37 along path $\mathcal{P}_2$. If these paths are stimulated at time zero, stimuli will enter cell **B** at times 7, 17, and 37.]

2.7.2.1 Odd delays and single delays. It is clear, however, that only even delays can be produced in this manner: the lengths of any two paths connecting two given points **A** and **B** differ by an even number. Yet the sequences of list (18) {as required by list (22)} preceded by a (stimulus) digit 1 {as required by list (21)} may contain digits 1 (i.e., stimuli) at odd distances. This conflict must be resolved. The resolution can be effected in various ways.

First, the principle of even difference of lengths between paths connecting the same two points depends for its validity on the crystal

lattice used. It holds for the quadratic lattice that we use here, but it fails for some other lattices. We might therefore consider changing the crystal lattice.

Second, we could lengthen the sequences of list (18) by inserting (no stimulus) digits 0, so as to make all the distances between the (stimulus) digits 1 {including the prefatory 1 required by rule (21)} even. This would increase the number of the subsequences encountered in building up the sequences of list (18), i.e., the number of the subsequences according to list (19). Thus there would have to be more sensitized states.

Third, we can introduce the odd delays directly. It is clearly sufficient to introduce a single delay. This means that we need an excited state, which occurs at time t, not after a stimulation at time $t - 1$ (as in all cases so far considered, i.e., for **T**, **T**′, **C**), but after a stimulation at time $t - 2$.

2.7.2.2 Single delays through the confluent states. A closer inspection indicates that, of the three alternatives described in Section 2.7.2.1, the last one is most convenient, and particularly most economical with respect to the number of new states required. In this connection two additional remarks are in order.

First, it suffices to supply a single delay mechanism for ordinary (**T**) stimuli. Indeed, special (**T**′) stimuli can be obtained from the ordinary ones by a fixed delay conversion process {namely (16)}. That is, if a rigidly timed sequence (in the sense of Sec. 2.7.1) of special (**T**′) stimuli is wanted, it is practical to produce it first (with a delay system according to Sec. 2.7.2.1) with ordinary (**T**) stimuli, and then convert from ordinary to special {with the help of rule (16)}.

Second, in order to introduce a single delay for ordinary stimuli, it is not necessary to introduce a new kind of state; it suffices to utilize and expand an existing kind. Indeed, it is quite practical to attribute this trait to **C**, because the other necessary functions of **C** need not be impaired by such a change.

That this is so will become clear in the course of our actual uses of **C**. For the present we will limit ourselves to formulating those changes in the rules that govern the behavior of **C** which are needed to introduce the desired (single) delay.

At present the excitation of **C** is described by rule (11). The stimulating effect of **C** is described by rules (12) and (16); the "killing" of **C** (i.e., its transfer into **U**) is described by rule (14).

Under these rules **C** has the states $\mathbf{C}_\epsilon$, where the index $\epsilon = 0, 1$ describes the present state of excitation. If the excitation is to be delayed by a single time unit, then **C** must remember for that length

of time what its state of excitation will be next. Hence two indices, e.g., ϵ, ϵ', will be needed. The states will be $\mathbf{C}_{\epsilon\epsilon'}$, where the index $\epsilon = 0, 1$ describes the present state of excitation, and the index $\epsilon' = 0, 1$ describes the next state of excitation. The effects on rule (11), and on rules (12), (14), and (16) are therefore the following.

In rule (11): where $\mathbf{C}_\epsilon$ (for $t - 1$) was transferred into $\mathbf{C}_{\epsilon''}$ (for t), now $\mathbf{C}_{\epsilon\epsilon'}$, (for $t - 1$) should be transferred into $\mathbf{C}_{\epsilon'\epsilon''}$ (for t).

In rules (12), (14), and (16): the role of $\mathbf{C}_\epsilon$ (for $t - 1$) is taken over by $\mathbf{C}_{\epsilon\epsilon'}$, (for $t - 1$).

Furthermore, it is natural that in the list (17) the (quiescent) state $\mathbf{C}_0$ should be replaced by the (completely quiescent) state $\mathbf{C}_{00}$.

The rigorous statements of the necessary modifications are accordingly these:

(23)
In rule (11) replace $n_\vartheta^{t-1} = \mathbf{C}_\epsilon$ by $n_\vartheta^{t-1} = \mathbf{C}_{\epsilon\epsilon'}$,
replace $n_\vartheta^{t} = \mathbf{C}_1$ by $n_\vartheta^{t} = \mathbf{C}_{\epsilon'1}$, and
replace $n_\vartheta^{t} = \mathbf{C}_0$ by $n_\vartheta^{t} = \mathbf{C}_{\epsilon'0}$.
In rules (12) and (16) replace $n_{\vartheta'}^{t-1} = \mathbf{C}_1$ by $n_{\vartheta'}^{t-1} = \mathbf{C}_{1\epsilon'}$;
in rule (14) replace $n_\vartheta^{t-1} = \mathbf{C}_\epsilon$ by $n_\vartheta^{t-1} = \mathbf{C}_{\epsilon\epsilon'}$.
In list (17) replace $\mathbf{C}_0$ by $\mathbf{C}_{00}$.

To conclude, we observe that with this method of introducing a (single) delay we replace the two states $\mathbf{C}_\epsilon$ by the four states $\mathbf{C}_{\epsilon\epsilon'}$; that is, we introduced two new states.

2.8 Summary

2.8.1 Rigorous description of the states and of the transition rule. We can now give a rigorous summary, i.e., a complete list of states and an exhaustive transition rule.

Let us write $\mathbf{T}_u$, $u = 0, 1$, in place of $\mathbf{T}$, $\mathbf{T}'$, respectively. Correspondingly, let us call the ordinary stimuli and the special stimuli *stimuli* [*0*] and *stimuli* [*1*], respectively.

The enumeration of states becomes this:

(S) The *states:*

The states are the following ones:

The *transmission states* $\mathbf{T}_{u\alpha\epsilon}$, where $u = 0, 1$ correspond to *ordinary* and *special;* $\alpha = 0, 1, 2, 3$ to *right, up, left, down;* $\epsilon = 0, 1$ to *quiescent* and *excited.*

The *confluent states* $\mathbf{C}_{\epsilon\epsilon'}$, where $\epsilon = 0, 1$ correspond to *quiescent* and *excited;* $\epsilon' = 0, 1$ to *next quiescent* and *next excited.*

The *unexcitable state* $\mathbf{U}$.

The *sensitized states* $\mathbf{S}_\Sigma$—where Σ has the range

(S.1) $$\Sigma = \theta, 0, 1, 00, 01, 10, 11, 000.$$

In addition, the $\mathbf{S}_\Sigma$ with

(S.2) $$\Sigma = 0000, 0001, 001, 010, 011, 100, 101, 110, 111,$$

in this order, are identified with

(S.3) $$\mathbf{T}_{u\alpha 0}\ (u = 0, 1;\ \alpha = 0, 1, 2, 3) \text{ and } \mathbf{C}_{00}$$

in the order $\mathbf{T}_{000}$, $\mathbf{T}_{010}$, $\mathbf{T}_{020}$, $\mathbf{T}_{030}$, $\mathbf{T}_{100}$, $\mathbf{T}_{110}$, $\mathbf{T}_{120}$, $\mathbf{T}_{130}$, $\mathbf{C}_{00}$. This is a total of 16 (transmission) + 4 (confluent) + 1 (unexcitable) + 8 (sensitized) = 29 states. Hence

(24) $$N = 29,$$

and the symbols $\mathbf{T}_{u\alpha\epsilon}$ $(u = 0, 1;\ \alpha = 0, 1, 2, 3;\ \epsilon = 0, 1)$, $\mathbf{C}_{\epsilon\epsilon'}$ $(\epsilon = 0, 1;\ \epsilon' = 0, 1)$, $\mathbf{U}$, $\mathbf{S}_\Sigma$ {Σ according to (S.1)} will be used in place of the 29 number values in expression (6) (cf. the remark after expressions (6) and (7) in Sec. 2.1.2).

Let us now consider the transition rule. First, note that the number of possibilities for this rule (i.e., for the function $\mathbf{F}$ in the sense of Sec. 2.1.2) is, according to expression (9) with $N = 29$,

(25) $$29^{(29^5)} \approx 10^{30,000,000}$$

(with three significant figures in the exponent). Second, the rules (10)–(16) and (21)–(23) constitute together the transition rule, and they can be summarized as follows.

(T) *The transition rule:*

(T.1)
Assume $n_\vartheta^{t-1} = \mathbf{T}_{u\alpha\epsilon}$.

(α) $n_\vartheta^{t} = \mathbf{U}$ if and only if $n_{\vartheta'}^{t-1} = \mathbf{T}_{u'\alpha'1}$, for some ϑ' with $\vartheta - \vartheta' = v^{\alpha'}$, and also $u \neq u'$.

(β) $n_\vartheta^{t} = \mathbf{T}_{u\alpha 1}$ if and only if (α) does not hold and either (a) or (b) holds:

(a) $n_{\vartheta'}^{t-1} = \mathbf{T}_{u\alpha'1}$ for some ϑ' with $\vartheta - \vartheta' = v^{\alpha'} \neq -v^{\alpha}$.

(b) $n_{\vartheta'}^{t-1} = \mathbf{C}_{1\epsilon'}$ for some ϑ' with $\vartheta - \vartheta' = v^{\beta} \neq -v^{\alpha} (\beta = 0, \cdots, 3)$.

(γ) $n_\vartheta^{t} = \mathbf{T}_{u\alpha 0}$ if and only if neither (α) nor (β) holds.

(T.2)
Assume $n_\vartheta^{t-1} = \mathbf{C}_{\epsilon\epsilon'}$.
(α) $n_\vartheta^{t} = \mathbf{U}$ if and only if $n_{\vartheta'}^{t-1} = \mathbf{T}_{1\alpha'1}$, for some ϑ' with $\vartheta - \vartheta' = \nu^{\alpha'}$.
(β) $n_\vartheta^{t} = \mathbf{C}_{\epsilon'1}$ if and only if (α) does not hold and both (a) and (b) hold:
(a) $n_{\vartheta'}^{t-1} = \mathbf{T}_{0\alpha'1}$ for some ϑ with $\vartheta - \vartheta' = \nu^{\alpha'}$.
(b) Never $n_{\vartheta'}^{t-1} = \mathbf{T}_{0\alpha'0}$ for an ϑ' with $\vartheta - \vartheta' = \nu^{\alpha'}$.
(γ) $n_\vartheta^{t} = \mathbf{C}_{\epsilon'0}$ if and only if neither (α) nor (β) holds.

(T.3)
Assume $n_\vartheta^{t-1} = \mathbf{U}$.
(α) $n_\vartheta^{t} = \mathbf{S}_\theta$ if and only if $n_{\vartheta'}^{t-1} = \mathbf{T}_{u\alpha'1}$, for some ϑ' with $\vartheta - \vartheta' = \nu^{\alpha'}$.
(β) $n_\vartheta^{t} = \mathbf{U}$ if and only if (α) does not hold.

(T.4)
Assume $n_\vartheta^{t-1} = \mathbf{S}_\Sigma$ {with Σ according to list (S.1)}.
(α) $n_\vartheta^{t} = \mathbf{S}_{\Sigma 1}$ if and only if $n_{\vartheta'}^{t-1} = \mathbf{T}_{u\alpha'1}$, for some ϑ' with $\vartheta - \vartheta' = \nu^{\alpha'}$.
(β) $n_\vartheta^{t} = \mathbf{S}_{\Sigma 0}$ if and only if (α) does not hold.

2.8.2 Verbal summary. The rigorous summary of Section 2.8.1 is a strict restatement of the verbal formulations and conclusions arrived at in Sections 2.2–2.7. In this sense, a verbal statement of the strict and formalistic contents of Section 2.8.1 is available in those sections. However, it seems desirable to give at this point a verbal restatement of the contents of Section 2.8.1, i.e., of its descriptions of the states and of the transition rule. Indeed, the formalism of Section 2.8.1 is not easy to follow without the verbal motivations of Sections 2.2–2.7. On the other hand, the verbal elaborations of those sections are lengthy and were arrived at stepwise. A direct verbal restatement is therefore indicated.

Such a restatement, covering the states and the transition rule together, is given in what follows.

There exist 16 *transmission states* $\mathbf{T}_{u\alpha\epsilon}$. The index u indicates the *class* of the state: $u = 0$ for *ordinary* and $u = 1$ for *special*. The index α indicates the *orientation* of the state: $\alpha = 0$ for *right*, $\alpha = 1$ for *up*, $\alpha = 2$ for *left*, and $\alpha = 3$ for *down*. The index ϵ indicates the present state of *excitation:* $\epsilon = 0$ for *quiescent*, and $\epsilon = 1$ for *excited*. A transmission state has one *output* direction and three *input* directions: the former is the direction defined by its orientation; the latter are all the others. A transmission state can be excited with a *delay 1*, by any immediately neighboring excited transmission state of the

same class, provided that the former lies in the output direction of the latter, which in turn must be lying in one of the former's input directions.

There exist four *confluent states* $\mathbf{C}_{\epsilon\epsilon'}$. The index ϵ indicates the present state of excitation, the index ϵ' indicates the next state of excitation: ϵ or $\epsilon' = 0$ for *quiescent*, and ϵ or $\epsilon' = 1$ for *excited*. The confluent states are viewed as being of class 0. For a confluent state, all directions are available both for inputs and for outputs. [But at any given time a direction cannot be used for both an input and an output.] A confluent state can be excited with a *delay 2* by those immediately neighboring transmission states of its own class (i.e., 0) in whose output direction it lies. The excitation will take place if there exists at least one such immediate neighbor, and if all such immediate neighbors (whatever their number) are excited.

A transmission state (of either class) can also be excited with a *delay 1* by any immediately neighboring excited confluent state, provided that the latter lies in one of the former's input directions.

There exists an unexcitable state **U**. This state is viewed as quiescent. Any transmission or confluent state is *killed* (i.e., transferred into the unexcitable state) by any immediately neighboring excited transmission state of the opposite class, provided that the former lies in the latter's output direction.

All the above states (transmission and confluent) go into their quiescent forms when no excitation or kill, according to the above rules, is provided for.

There exist eight *sensitized states* $\mathbf{S}_\Sigma$, with Σ according to list (S.1). We will also use the symbol $\mathbf{S}_\Sigma$ with Σ according to list (S.2), but these latter states are not considered to be sensitized ones. They are identified with the quiescent transmission and quiescent confluent states according to list (S.3). (For the references to lists (S.1)–(S.3), cf. Sec. 2.8.1.) A sensitized state $\mathbf{S}_\Sigma$ will in any case undergo a change (immediately, i.e., with a *delay 1*), namely into $\mathbf{S}_{\Sigma 0}$ or $\mathbf{S}_{\Sigma 1}$. The change into $\mathbf{S}_{\Sigma 1}$ will take place under the influence of any immediately neighboring excited transmission state (of either class), provided that the sensitized state lies in the output direction of the latter. Otherwise, a change into $\mathbf{S}_{\Sigma 0}$ will take place.

We re-emphasize that this last mentioned rule applies only as long as the state is sensitized, i.e., as long as Σ is according to list (S.1), and not according to list (S.2) (i.e., as long as it has not reached a maximum size).

[*2.8.3 Illustrations of the transition rule.* Each cell of von Neumann's infinite cellular structure is occupied by the same 29-state finite

automaton. As explained in Section 1.3.3.5, an "infinite cellular automaton" consists of this infinite cellular structure together with an initial cell assignment. An "initial cell assignment" is a *finite* list of cells together with an assignment of a state to each cell of the list; all cells not in the list are assigned the unexcitable state **U**. An initial cell assignment determines the state of an infinite cellular automaton at time zero. The history of the infinite cellular automaton is then determined by the transition rule, which gives the state of each 29-state finite automaton at time $t + 1$ as a function of its own state and the states of its four immediate neighbors at time t.

The 29 states and their transition rule are summarized in Figures 9 and 10. Von Neumann also symbolized pairs of transmission states by a zero (for ordinary) or a one (for special), together with an arrow indicating the output direction. For example, the pair of states $\mathbf{T}_{00\epsilon}$ is represented by $\overset{0}{\rightarrow}$, while $\mathbf{T}_{11\epsilon}$ is represented by $\uparrow 1$.

There are many ways of viewing the 29-state automaton which occupies a single cell of von Neumann's cellular structure. It can be viewed as a finite automaton which is constructed from switch elements and delay elements and which is connected to its four immediate neighbors by wires crossing its four boundaries. In what follows it is more fruitful to view it as a set of primitive elements (sub-automata) together with control apparatus for switching back and forth among these elements. This amounts to partitioning the 29-states of a cell into subsets which correspond to certain functions.

Consider as an example the pair of transmission states $\mathbf{T}_{000}$ and $\mathbf{T}_{001}$, i.e., $\mathbf{T}_{00\epsilon}$ for $\epsilon = 0, 1$. This pair of states functions as a disjunction (+ neuron) feeding a unit delay whose output is directed to the right. The potential inputs of $\mathbf{T}_{00\epsilon}$ are from its immediate neighbors above it, to its left, and below it. These inputs come from confluent states or from other ordinary transmission states which are directed toward it. Thus in Figure 11a, the cell $\mathbf{T}_{00\epsilon}$ behaves as follows. If either $\mathbf{C}_{1\epsilon'}$ or $\mathbf{T}_{011}$ at t, then $\mathbf{T}_{001}$ at $t + 1$; if both $\mathbf{C}_{0\epsilon'}$ and $\mathbf{T}_{010}$ at t, then $\mathbf{T}_{000}$ at $t + 1$. In other words, as long as Figure 11a is limited to the states shown there, the two states $\mathbf{T}_{00\epsilon}$ can be thought of as constituting a disjunctive element with inputs from the left and below and an output (after unit delay) to the right (at f).

The set of four confluent states $\mathbf{C}_{\epsilon\epsilon'}$ ($\epsilon, \epsilon' = 0, 1$) perform the functions of conjunction ("and", "·"), double delay, wire branching (splitting), and conversion of ordinary stimuli into special stimuli; von Neumann also symbolized these four states by **C**. These states have no direction, the direction of their functioning being determined by the directions of the transmission states (both ordinary and

special) in the four immediately neighboring cells. Consider a set of states $\mathbf{C}_{\epsilon\epsilon'}$ occupying a given cell. The inputs to this set $\mathbf{C}_{\epsilon\epsilon'}$ are from ordinary transmission states directed toward it. The outputs from this set $\mathbf{C}_{\epsilon\epsilon'}$ are to both ordinary and special transmission states not directed toward it. See Figures 11b and 11c. In Figure 11b the cell $\mathbf{C}_{\epsilon\epsilon'}$ behaves as follows. If both $\mathbf{T}_{001}$ and $\mathbf{T}_{011}$ at t, then $\mathbf{C}_{\epsilon'1}$ at $t + 1$ and $\mathbf{C}_{1\epsilon''}$ at $t + 2$; if either $\mathbf{T}_{000}$ or $\mathbf{T}_{010}$ at t, then $\mathbf{C}_{\epsilon'0}$ at $t + 1$ and $\mathbf{C}_{0\epsilon''}$ at $t + 2$. In other words, in the context of Figure 11b, the four states $\mathbf{C}_{\epsilon\epsilon'}$ can be thought of as constituting a conjunctive element with inputs from the left and below and with output (after two units of delay) to the right (at g).

It is convenient to think of the state ϵ of a $\mathbf{T}_{u\alpha\epsilon}$ cell or a $\mathbf{C}_{\epsilon\epsilon'}$ cell as its output state at a given time, and to think of the composite state of its immediate neighbors as its input state at that time. It is also convenient to use short bars to indicate an input to a cell (a, b, c, d, and e of Fig. 11) or as output from a cell (f, g, h, i, and j of Fig. 11). Under this convention, we have for Figure 11:

$$\begin{aligned} f(t+3) &= [a(t) + b(t+1)] \\ g(t+3) &= [c(t) \cdot d(t)] \\ h(t+4) &= e(t) \\ i(t+5) &= e(t) \\ j(t+4) &= e(t). \end{aligned}$$

All the outputs are ordinary stimuli except that the output j is a special stimulus.

Each switching function (truth function) of disjunction (+) and conjunction ($\cdot$) can be realized, with suitable delay, by a cellular network of ordinary states ($\mathbf{T}_{0\alpha\epsilon}$ and $\mathbf{C}_{\epsilon\epsilon'}$). Storage loops may be made of ordinary states. In Figure 12 the loop *B1*, *B2*, *A2*, *A1* stores the sequence 10000, which will cycle around the square ad infinitum, continually feeding into cell *C1*. Negation is not represented directly in the 29 states but is synthesized from the destructive (reverse) and constructive (direct) processes. See Section 3.2.2 and Figure 17 below.

The direct process changes a cell from the unexcitable state **U** into one of the nine quiescent states $\mathbf{T}_{u\alpha0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$) or $\mathbf{C}_{00}$. Transmission states (ordinary or special) directed toward **U** initiate and control the direct process, and sensitized states serve as intermediaries. Any (or several) $\mathbf{T}_{u\alpha1}$ directed toward **U** converts it to $\mathbf{S}_\theta$. Thereafter $\mathbf{S}_\Sigma$ is followed by (a) $\mathbf{S}_{\Sigma1}$ if some $\mathbf{T}_{u\alpha1}$ is directed toward the cell, (b) $\mathbf{S}_{\Sigma0}$ otherwise, until the direct process terminates in a $\mathbf{T}_{u\alpha0}$ or $\mathbf{C}_{00}$ in accordance with Figure 10. For example, when sent into a cell which is in state **U**, the sequence 10000 produces

$\mathbf{T}_{000}$, the sequence 1011 produces $\mathbf{T}_{100}$, and the sequence 1111 produces $\mathbf{C}_{00}$.

The direct process is illustrated in Figure 12. As indicated by the subscripts (values of ϵ and ϵ'), the loop *B1*, *B2*, *A2*, *A1*, stores the sequence 10000 in that order at time zero. This sequence will cycle around the loop ad infinitum, repeatedly feeding into cell *C1*. It will pass through cell *C1* with a unit delay. Its effect on cell *D1* through time is as follows:

Time:	0	1	2	3	4	5	6	7	$\cdots$
Input to *D1*:	0	1	0	0	0	0	1	0	$\cdots$
State of *D1*:	**U**	**U**	$\mathbf{S}_{\theta}$	$\mathbf{S}_{0}$	$\mathbf{S}_{00}$	$\mathbf{S}_{000}$	$\mathbf{T}_{000}$	$\mathbf{T}_{001}$	$\cdots$.

The process will now repeat, with cell *D1* feeding the sequence 10000 into the cell on its right, causing it to pass through the sequence $\mathbf{S}_{\theta}$, $\mathbf{S}_{0}$, $\mathbf{S}_{00}$, $\mathbf{S}_{000}$, $\mathbf{T}_{000}$, $\mathbf{T}_{001}$, $\cdots$. This process will again repeat, ad infinitum. Thus an infinite cellular automaton which has the first pattern of Figure 12 as its initial cell assignment will grow an ever lengthening communication channel $\overset{0}{\rightarrow}\overset{0}{\rightarrow}\overset{0}{\rightarrow}\cdots$ to the right.

The reverse process (destruction, killing) changes any transmission state $\mathbf{T}_{u\alpha\epsilon}$ and any confluent state $\mathbf{C}_{\epsilon\epsilon'}$ into **U**. An excited ordinary transmission state $\mathbf{T}_{0\alpha 1}$ kills a special transmission state $\mathbf{T}_{1\alpha\epsilon}$ toward which it is directed, and an excited special transmission state $\mathbf{T}_{1\alpha 1}$ kills an ordinary transmission state $\mathbf{T}_{0\alpha\epsilon}$ or confluent state $\mathbf{C}_{\epsilon\epsilon'}$ toward which it is directed. Examples are given in Figure 13. In Figures 13a and 13b cells shown with question marks at time 1 are so marked because their states at this time depend on the states of the cells in the surrounding environment. Except for Figures 13a and 13b, we always assume that the finite cellular arrays of our figures are not affected by the direct or reverse process operating on them from the environment. Note that the special and ordinary transmission states of Figure 13b kill each other.

The reverse process dominates reception. If a "kill" stimulus enters a cell at time t, that cell will be in state **U** at time $t + 1$, no matter what other stimuli may have entered the cell at time t. On the other hand, killing does not dominate emission, in the sense that if a cell is prepared to emit a stimulus at time t it will in fact emit that stimulus, even though it also receives a kill stimulus at time t. We will illustrate these facts with Figure 13c.

Assume in Figure 13c that ordinary stimuli enter input a at times 0 and 1, a special stimulus enters input c at time 1, and no other stimuli enter the figure at any other time. The ordinary stimulus entering a at time 0 enters cell *B1* at time 1, leaves cell *B1* at time 2, and leaves

cell *C1* (output *b*) at time 3. The special stimulus which enters input *c* at time 1 enters cell *B1* at time 2, causing cell *B1* to be in state **U** at time 3. The ordinary stimulus entering *a* at time 1 enters cell *B1* at time 2, but, since the special kill stimulus also enters cell *B1* at time 1, the ordinary stimulus has no effect. Hence the first ordinary stimulus into input *a* is transmitted through cells *A1*, *B1*, *C1* and is emitted from output *b*, but the second ordinary stimulus into input *a* is dominated by the kill stimulus from input *c* and is lost.

One fundamental construction in the cellular system is to change the state of a remote cell by the direct process and then wipe out the constructing path used for the construction. The technique is illustrated in Figure 14. Ordinary stimuli are fed alternately into inputs *i* and *j*, they travel along the path *B2-D2*, and cell *D3* is left in the quiescent state. The sequences required are as follows (cf. Fig. 10). (a) Input *i* is supplied with 10000, 10000, 1010, and 1111. The path *B2–D2* becomes an ordinary transmission channel, and cell *D3* is left in the desired state **C**, as in Figure 14b. (b) Input *j* is supplied with 1,1011; 1,1011; and 1. The cells *B2* and *C2* become **U**, $\mathbf{S}_{\theta}$, $\mathbf{S}_{0}$, $\mathbf{S}_{01}$ and $\mathbf{S}_{011}$ (i.e., $\mathbf{T}_{100}$) in turn. The cell *D2* becomes **U**, as in Figure 14c. (c) The sequence 1,10000, and 1 into *i* produces Figure 14d. (d) Finally, the single stimulus 1 into *j* produces Figure 14e. Thus we are left with cell *D3* in the desired state and with the constructing path *B2–D2* in the unexcitable state.

The transformations of Figure 14 require 37 time steps, measuring time from the input sides of cell *B2*. As described above, stimuli come from inputs *i* and *j* alternately, but, since the absence of a stimulus is represented by "0," we can think of a sequence of length 37 going into cell *B2* via input *i* and a simultaneous sequence of length 37 going into cell *B2* via input *j*. The path from input *j* to cell *B2* requires 2 more units of time than the path from input *i* to cell *B2;* hence the input to *j* must lead the input to *i* by 2 units of time. Consequently, the transformation from Figure 14a to Figure 14e will be brought about by the two 36-bit sequences of ordinary stimuli shown at the bottom of Figure 14; in these sequences time goes from left to right. The sequence for input *i* may be obtained by prefixing *i* with a sequence of 36 cells in states $\mathbf{T}_{00\epsilon}$, each ϵ being chosen according to the needed input into *i*. Likewise, the sequence for input *j* may be obtained by prefixing *j* with a sequence of 36 cells in the appropriate $\mathbf{T}_{00\epsilon}$ states.

The above technique may be used to construct an arbitrary finite array $\mathfrak{A}$ of quiescent cells, i.e., cells in the states **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), or $\mathbf{C}_{00}$. For each such array $\mathfrak{A}$, there are two binary

sequences of stimuli which, when fed into inputs i and j of Figure 14, will produce the array $\mathfrak{A}$ to the right. Moreover, these two sequences will emanate from two linear arrays of cells in the states $\mathbf{T}_{00\epsilon}$, with the ϵ's properly chosen. These two linear arrays, together with cells *A1*, *A4*, *B1–B4* of Figure 14 constitute an array $\mathfrak{B}$. Thus we have the following result about constructions in von Neumann's cellular structure: for each quiescent finite array $\mathfrak{A}$, there is a finite array $\mathfrak{B}$ and a time τ such that array $\mathfrak{A}$ will appear at time τ in the infinite cellular automaton which has $\mathfrak{B}$ as its initial cell assignment. Moreover, the limitation that the cells of array $\mathfrak{A}$ be quiescent, while important (cf. Sec. 1.6.3.2 above), is not serious, for the array $\mathfrak{B}$ can impart a starting stimulus to array $\mathfrak{A}$ before returning the constructing path to the unexcitable state.

It is clear from the previous construction that area $\mathfrak{B}$ is always larger (contains more cells) than $\mathfrak{A}$. Compare Section 1.6.1.1 above. Von Neumann circumvents this difficulty by designing a universal constructing automaton **D** and attaching to it a tape description of itself $\mathbf{L}_\mathbf{D}$; see Section 1.6.1.2 above. This universal constructing automaton will have, among other powers, the power of a universal Turing machine. The technique of Figure 14 will play an essential role in the operation of this universal constructing automaton **C**. See Chapters 4 and 5 below.]

DESIGN OF SOME BASIC ORGANS

3.1 Introduction

3.1.1 Free and rigid timing, periodic repetition, and phase stop. The organs that will be constructed in the sections that follow will exhibit two relevant traits (separately or jointly): *free timing* and *rigid timing*.

We have *free timing* when the emission of certain stimuli (at a specified point and in a certain order) is provided for, but the time intervals or *delays* from one such emission to the next are not prescribed.

We have *rigid timing* when the delays of the above pattern are numerically specified. We may alternatively express this by stating that these delay periods are filled with "no stimuli." (Clearly there will be $d - 1$ of these for a delay period between a prescribed stimulus and its successor, if the specified delay between these two is d.) The rigidly timed sequence of stimuli can therefore also be described as a "stimulus-no-simulus sequence" of uninterruptedly consecutive events. In this form it will be designated by a sequence of 0's and 1's, each 1 standing for a "stimulus," and each 0 for a "no stimulus": $\overline{i^1 \cdots i^n}$ (each $i^\nu = 0, 1$). Whenever this notation $\overline{i^1 \cdots i^n}$ is used, rigid timing is implied.

[The direct process of Section 2.6.3.2 above involves rigid timing. As von Neumann said there, in rigid timing "the absence of a stimulus has as definite an effect as the presence of one, and therefore the stimuli that are required must be delivered at definite times, without any delays other than the expressly required ones."

As we saw in Section 1.3.3.5, there is no bound on the number of cells which can be in the excited state at any given time, though of course this number is always finite. Thus von Neumann's cellular structure allows for an indefinite amount of parallelism. But in designing his self-reproducing automaton von Neumann did not make much use of the potential parallelism of his cellular structure. Rather, his self-reproducing automaton works like a serial digital computer,

with most organs normally quiescent. In this respect it is similar to the EDVAC (see pp. 9–11 above).

When an organ of the self-reproducing automaton is stimulated by a sequence containing one or more stimuli, the organ will produce an output (response) after a delay. Usually the delay between input and output doesn't matter, because no other construction or computation is going on simultaneously. Likewise, the delay between one input to an organ and the next input to that organ does not usually matter, provided that this delay is sufficiently great for the organ to finish one action before starting the next. The main exception to these statements concerns the tape unit of Chapter 4 below. The method von Neumann used for lengthening and shortening the tape requires simultaneous (parallel) action in two connecting loops.]

By a *periodic repetition* of such a sequence $\overline{i^1 \cdots i^n}$ a rigidly timed periodic repetition is meant, i.e., one with uninterruptedly consecutive periods $\overline{i^1 \cdots i^n}$, unless the opposite is expressly stated. Thus a periodically repeated $\overline{i^1 \cdots i^n}$, to be designated $\overline{\overline{i^1 \cdots i^n}}$, means $\overline{i^1 \cdots i^n i^1 \cdots i^n i^1 \cdots i^n \cdots \cdots}$. Such periodic repetitions are never intended to go on indefinitely. By a *stop in phase* k $(= 1, \cdots, n)$ in *period* s $(= 1, 2, \cdots)$, we mean that the sequence is interrupted immediately before the i^k in the s-th period (i.e., no stimuli are emitted at this time or later). We may also call this a *stop* at *step* ℓ with $\ell = nx + k$ $(\ell = 1, 2, \cdots)$. By a *stop* in *period* s (without stating a phase) we mean one in phase 1 (i.e., at step $ns + 1$).

3.1.2 Construction of organs, simple and composite. We will now construct successively certain organs, going from the simpler to the more complicated. Most of these organs will be composites of previously defined organs with each other and with suitable controlling and connecting networks. Each one of the organs to be defined will be given a name and a symbol, so that it can be identified when it occurs as a part of a subsequent larger, composite organ.

[In some cases von Neumann gave algorithms for designing any organ of a given class; this is so for pulsers (Sec. 3.2), decoding organs (Sec. 3.3), and coded channels (Sec. 3.6). In the case of the triple-return counter (Sec. 3.4) and the $\overline{1}$ vs. $\overline{10101}$ discriminator (Sec. 3.5), he designed specific organs. To make von Neumann's algorithms and constructions easier to follow, we discuss a completed organ at the beginning of each section.

Since von Neumann was interested in an existence proof of self-reproduction he did not, in general, attempt to minimize his design.]

3.2 Pulsers

3.2.1 The pulser: structure, dimensions, and timing.

[Figure 15 shows two pulsers designed according to von Neumann's algorithm. We will explain how they work.

A stimulus (e.g., $\mathbf{T}_{001}$) into input a of pulser $\mathbf{P}(\overline{111})$ at time t is split at each confluent element and produces a sequence $\overline{111}$ from output b at times $t + 9$, $t + 10$, and $t + 11$. The cross-hatched cell is in the unexcitable state **U**.

The characteristic $\overline{10010001}$ contains three ones, so three paths (B, D, and F) are needed for $\mathbf{P}(\overline{10010001})$, with relative delays 0, 3, and 7, respectively. The confluent cells along the bottom produce relative delays of 0, 1, and 2, respectively. Additional delays are achieved in two ways. Replacing two ordinary-up transmission states by a block like *C5*, *C6*, *D5*, and *D6* adds two units of delay to a path; by this means we get relative delays of 0, 3, and 6 in the three paths. Replacing an ordinary transmission state in row *2* by a confluent state (e.g., *F2*) adds a unit of delay to a path; in this way we obtain relative delays of 0, 3, and 7 in the three paths B, D, and F, respectively.]

We begin with the *pulser*. This organ has an input a and an output b. Upon a stimulus at a it will emit a prescribed sequence $i^1 \cdots i^n$ at b.

The relation between the stimulus at a and the response at b is freely timed; i.e., the delay between these is not prescribed at this point (cf., however, the remarks at the end of this subsection). This pulser has the symbol $\mathbf{P}(\overline{i^1 \cdots i^n})$. The sequence $\overline{i^1 \cdots i^n}$, which can be prescribed at will, is its *characteristic*; n is its *order*.

The principles involved in constructing a pulser are quite simple. The actual network is successively developed in Figure 16. This construction and the network that results from it will be discussed in the balance of this section.

Let $\nu_1, \cdots, \nu_k$ be the ν for which $i^\nu = 1$, i.e., the positions of the stimuli in the sequence $\overline{i^1 \cdots i^n}$. Then $1 \leqq \nu_1 < \nu_2 < \cdots < \nu_k \leqq n$. Hence $\nu_h \geqq h$. Write

$$(1') \qquad \begin{cases} \nu_h - h = 2\mu_h + r_h, \\ \text{where } \mu_h = 0, 1, 2, \cdots; r_h = 0, 1. \end{cases}$$

The input stimulus at a must be broken up, or rather multiplexed into k distinct stimuli, numbers $h = 1, \cdots, k$, arriving at b with the relative delays ν_h ($h = 1, \cdots, k$), respectively.

Consider first the network of Figure 16a. A stimulus arriving at a

has k paths available to reach b: on path number $h (h = 1, \cdots, k)$ it goes horizontally from a to the **C** number h, then it turns vertically up, ascends to the top line, and continues there horizontally to b. There are $(2h - 1) + u + (2(k - h) + 1) = 2k + u$ steps, but, since h of them are **C**'s, the entire delay involved is $(2k + u) + h$. That is, path number h has the relative delay (relatively to the other paths!) h.

Note that this network has the width $2k - 1$, but that its height $u + 2$ is still undetermined, along with u.

We must now replace the relative delay h of the path number h in Figure 16a by a relative delay ν_h (for each $h = 1, \cdots, k$); i.e., we must increase it by $\nu_h - h = 2\mu_h + r_h$ {cf. formula $(1')$}. We replace therefore each one of the k vertical branches in Figure 16a by a suitable delay network, say branch number h by a network N_h, as shown in Figure 16b. (We have also moved input a one step left, by placing a $\overset{0}{\rightarrow}$ before the first **C**.) Hence N_h must produce a delay $2\mu_h + r_h$.

N_h is shown again in Figure 16c, indicating its input c_h and its output d_h. (c_h is immediately above the **C** number h, d_h above this, under the top line, i.e., immediately below the $\overset{0}{\rightarrow}$ number $2h - 1$ in the top line.) To achieve the desired delay $2\mu_h + r_h$, N_h may be built up by stacking μ_h delay-two blocks vertically, plus a single delay-one block if $r_h = 1$. The former are shown in Figure 16d; their height is 2. The latter is shown in Figure 16e; its height is 1. The total height of N_h is then adjusted to the uniform value u by inserting an ordinary vertical ascent of length $u - 2\mu_h - r_h = u - (\nu_h - h) = s_h$ {cf. formula $(1')$}. This is shown in Figure 16f. (The delays referred to above are, of course, always relative ones, compared to an ordinary vertical ascent.) It is best to put the μ_h blocks of Figure 16d at the bottom of N_h, the (possible, single) block of Figure 16e at the top of N_h, and the vertical insertion of Figure 16f in between. The reason for this last precaution is that a **C** of Figure 16e must never be in contact with a transmission state of Figure 16d, since this might produce unwanted stimulations. The **U**-padding in Figures 16e and 16f is such that the contact in question could occur only between Figures 16e and 16d, and then only with the latter to the right of the former. Hence the top line of the N_h $(h = 1, \cdots, k)$ must contain no Figure 16d block immediately to the right of a Figure 16e block.

In view of all this we have, therefore, these conditions: always $u \geqq \nu_h - h$. If some $r_h = 1$ (i.e., $\nu_h - h$ odd, i.e., some Fig. 16e

block in the top line), and the maximum of $\nu_h - h$ is even, then $u > \nu_h - h$ for every h. In other words: $u \geqq u^0$, where

$$(2') \quad \begin{cases} u^0 = \underset{h=1,\cdots,k}{\text{Max}} (\nu_h - h) + \epsilon^0, \\ \text{where } \epsilon^0 = 1 \text{ if the Max is even but some } \nu_h - h \text{ is odd,} \\ \text{and } \epsilon^0 = 0 \text{ otherwise.} \end{cases}$$

It is, of course, simplest to put $u = u^0$.

[This rule is wrong when $\nu_k - k = 1$, as in $\mathbf{P}(\overline{1010})$, for a confluent state in the bottom row cannot directly drive a confluent state above it. This oversight may be corrected in different ways. We correct it by adding the following restriction to von Neumann's rule $(2')$:

$$\text{If } \nu_k - k = 1, \text{ then } u = 2.$$

Note that $\text{Max } (\nu_h - h) = \nu_k - k$.

Von Neumann introduced ϵ^0 because he did not want a Figure 16d to be immediately to the right of a Figure 16e, since the confluent state of the latter figure would affect an ordinary transmission state of the former figure. However, if Figure 16e is in the next to the top row of the pulser, and Figure 16d occupies this row and the row below it, this effect does not in fact change the output of the pulser. We could therefore replace von Neumann's design algorithm by a new algorithm with this simpler rule for determining u:

$$\text{If } \nu_k - k = 1, \text{ then } u = 2$$

$$\text{Otherwise, } u = \nu_k - k.$$

We will not make this replacement, however, since we wish to keep as close to von Neumann's original design as possible.]

This completes the construction. As Figure 16b shows, the area of this network has the width $2k$ and the height $u + 2$. An abbreviated representation of this network is given in Figure 16g.

Note that the sequence $\overline{i^1 \cdots i^n}$, i.e., the relative delays ν_h $(h = 1, \cdots, k)$, is achieved with a certain preliminary absolute delay. We saw that this was $2k + u$ under the conditions of Figure 16a. The insertion of the $\overset{0}{\rightarrow}$ before the first **C** in Figure 16b raises this to $2k + u + 1$. Thus in the final arrangement, represented by Figure 16g, a stimulus at a starts the sequence $\overline{i^1 \cdots i^n}$ after a preliminary [absolute] delay $2k + u + 1$ at b. (This means that the first stimulus or no-stimulus position of that sequence, i^1, has a delay $2k + u + 2$.)

If input a is stimulated several times, the emissions at b take place concurrently, irrespectively of whether the sequences $\overline{i^1 \cdots i^n}$ that

are so induced overlap or not. That is, there is no corruption by interference in this network.

[We will summarize the external characteristics of the pulser $\mathbf{P}(\overline{i^1 \cdots i^n})$. See Figure 16g.

The width of the pulser is $2k$, where k is the number of ones in the characteristic $\overline{i^1 \cdots i^n}$. Von Neumann implicitly assumed that $k \geqq 2$; for $k = 0$ or $k = 1$ no organ is needed.

The height of the pulser is $u + 2$, where u is defined as follows: $\nu_1, \cdots, \nu_k$ are the ν for which $i^\nu = 1$. Von Neumann's rule for u, as corrected, is

If $\nu_k - k = 1$, then $u = 2$.

Otherwise, $u = (\nu_k - k) + \epsilon^0$, where

$\epsilon^0 = 1$ if $(\nu_k - k)$ is even but some $\nu_h - h$ $(h = 1, \cdots, k)$ is odd,

$\epsilon^0 = 0$ otherwise.

Note that k is the number of ones in the characteristic and that ν_k is the superscript of the rightmost one; hence $\nu_k - k$ is the total number of zeros in the sequence preceding the rightmost one of the characteristic.

The delay between the input pulse going into a and the first output i^1 emerging from b is $2k + u + 2$.]

3.2.2 The periodic pulser: structure, dimensions, timing, and the $\mathbf{PP}(\overline{\overline{1}})$ *form.*

[Figure 17 shows two pulsers designed according to the algorithm von Neumann developed next.

The periodic pulser $\mathbf{PP}(\overline{\overline{10010001}})$ is constructed from a pulser $\mathbf{P}(\overline{10010001})$ which produces one occurrence of the desired sequence, a periodic repeater *G4–G6* and *H4–H6* of period eight, a mechanism *H1–H3* for turning the repeater off, and a pulser $\mathbf{P}(\overline{11111111})$ which produces the signals needed for turning the repeater off.

A "start" stimulus into input a_+ at time t will produce $\overline{10010001}$ from an output b' of $\mathbf{P}(\overline{10010001})$ at times $t + 29$ through $t + 36$. The sequence $\overline{10010001}$ is emitted repeatedly from output b until the repeater is turned off. A "stop" stimulus into input a_- will cause the top pulser to emit a sequence of 10 ones. These will pass through the confluent state *H2*, the special transmission state *H3*, and into cell *H4*. The first five pulses will induce the transformation $\mathbf{C}_{\epsilon\epsilon'} \rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_1 \rightarrow \mathbf{S}_{11} \rightarrow \mathbf{S}_{111}$ $(= \mathbf{C}_{00})$, and the last five pulses will repeat this transformation. No matter what the contents of the periodic repeater are, they will be wiped out. Note in this connection that since a sequence of ones is used to turn the repeater off, any ones entering cell *H4* from the left (i.e., from the repeater itself) are

ineffectual. This is a motive for putting the confluent state at the bottom of the periphery of the tree of Figure 10.

Since the periodic pulser **PP**($\overline{1}$) is used very often, von Neumann elected to simplify the design for this special case. Here he used the fact that, when Figure 17b is operating, there is a sequence of six ones in its periodic repeater. Once the transformation of the confluent state *E3* is begun by a pulse from *E2* the transformation is completed by the four ones remaining in the periodic repeater (two ones being lost when *E3* is killed). This periodic pulser does not, however, work in all contexts. See the editorial discussion at the end of this subsection.]

The next organ that we construct is the *periodic pulser*. This organ has two inputs a_+, a_-, and an output b. Upon a stimulus at a_+ it will begin to emit a prescribed, periodically repeated sequence, say $\overline{\overline{i^1 \cdots i^n}}$ at b. Upon a stimulus at a_- this emission at b will be stopped.

The relation between the stimuli at a_+ and a_- and the (starting or stopping of the) response at b is freely timed; i.e., the two delays between these are not prescribed at this point (cf., however, the remarks at the end of this section).

This periodic pulser has the symbol **PP**($\overline{\overline{i^1 \cdots i^n}}$). The sequence $\overline{\overline{i^1 \cdots i^n}}$, which can be prescribed at will, is its *characteristic*; n is its *order*.

The required network is successively developed in Figure 18. This construction and the network that results from it are discussed in the balance of this section.

The operation of producing the periodically repeated sequence $\overline{\overline{i^1 \cdots i^n}}$ is best decomposed into two suboperations. First, we produce a single sequence $\overline{i^1 \cdots i^n}$; second, we repeat it periodically. The first task calls for the pulser **P**($\overline{i^1 \cdots i^n}$) of Section 3.2.1, according to Figure 16g. We write a', b' for a, b there. Then our present input a_+ must either feed into or be the input a'. The second task requires attaching to the output b' an organ that will repeat any period of length n to the output b.

The simplest periodic repeater is a (closed) cycle of transmission states. The shortest cycle of this kind has a period 4, as shown in Figure 18a. Clearly any [such] cycle must have an even period; hence the general cycle has the period 2ℓ, $\ell = 2, 3, \cdots$. A cycle of this period is shown in Figure 18b.

Our present output b must issue from this repeater. Hence at least one of its cells must be able to stimulate in two directions (b, as well as its own successor in the cycle); i.e., it must be a **C**. The output b has two possible positions on this **C**: b_1, b_2.

This is shown in Figure 18c. The **C** raises the length of the period to $2\ell + 1$. This is an odd period. If an even period is wanted, a second **C** must be inserted (since the first one cannot be removed). This is shown in Figure 18d; the period length is here $2\ell + 2$. (Note that the two **C** are not immediate neighbors, since a **C** cannot stimulate a **C**.) Both alternatives are jointly represented by Figure 18e, where

$$\mathbf{X} \begin{cases} = \underset{\leftarrow}{0} \text{ for } r = 1. \\ = \mathbf{C} \text{ for } r = 2. \end{cases}$$

Here the period is $n = 2\ell + r$, where $\ell = 2, 3, \cdots$; $r = 1, 2$. That is, we can handle precisely the orders (periods) $n = 5, 6, \cdots$. Thus the order n is subject to the condition

$$(3') \qquad n \geqq 5.$$

If the condition (3′) is violated, i.e., if the order n is <5, then the period $\overline{i^1 \cdots i^n}$ may be repeated $\mathfrak{o}$ times. This replaces n by $\mathfrak{o}n$, and we need only choose $\mathfrak{o}$ so that $\mathfrak{o}n$ fulfills (3′), i.e., $\mathfrak{o}n \geqq 5$.

Let us now return to Figure 18e: The output b must be on the **C**, as shown there. The input of the cycle, i.e., the point fed by b', should be as close as possible to this **C**, to minimize the delay. It cannot be on the **C** itself, because **C**'s remaining free side (whichever of b_1, b_2 is not used for b) will be needed for another purpose. We place it therefore immediately before **C**, at b_1' or b_2'. (The sense of the cycle has been arranged to maximize this accessibility.)

Now the pulser $\mathbf{P}(\overline{i^1 \cdots i^n})$ of Figure 16g and the cycle of Figure 18e can be placed into actual contact. This is shown in Figure 18f, where

$$n = 2\ell + r;$$

$$\ell = 2, 3, \cdots; r = 1, 2.$$

$$\mathbf{X} \begin{cases} = {}^{\uparrow}\mathbf{0} \text{ for } r = 1. \\ = \mathbf{C} \text{ for } r = 2. \end{cases}$$

Figure 18f is drawn as if $\ell < u + 2$, but $\ell \geqq u + 2$ is equally possible. Note that the cycle is rotated against its previous form (Fig. 18e vs. Fig. 18f), to have a convenient fit with $\mathbf{P}(\overline{i^1 \cdots i^n})$. In addition the two organs are separated by a column of **U**'s, to avoid the possibility of unwanted stimulations due to contacts between a **C** of one organ and a transmission state of the other.

The network of Figure 18f takes care of starting $\overline{\overline{i^1 \cdots i^n}}$; there remains the task of providing a network for stopping $\overline{\overline{i^1 \cdots i^n}}$.

Since we have not introduced any states (or stimuli) for expressing the negation or inhibition directly, this must now be done by indirect means. The obvious way to achieve this is by using special stimuli, i.e., by changing the character of one of the transmission and confluent states that make up the cycle of Figure 18e (which is, after a rotation, part of Fig. 18f). This means that a_- should control a stimulus [1] which is directed against one of the exposed sides of the cycle.[1] Figure 18g shows the organ that will do this. Since it will be convenient to stimulate this organ (from a_- via a'') from above and to attach it to the upper side of the cycle, Figure 18g shows it in this position. The output b'' of this organ may be attached to any cell of the cycle. In the position shown (cf. Figs. 18f and 18g) this can be the $\overset{0}{\rightarrow}$ or the **C**; in the two other possible positions (on Fig. 18f, i.e., from the right or from below) it could be the **C**, any $\downarrow^0$, the $\overset{0}{\leftarrow}$ (from the right), the $\overset{0}{\leftarrow}$ or the **X** (from below).

With this arrangement, a stimulus [0] from a_-, arriving at a'', would excite **C** and then $\downarrow^1$, and deliver a stimulus [1] at b'' with a delay 3 (counted from a''). Hence, with a delay 4, that cell of the cycle which is in contact with b'' will go into the state **U** and cease to emit. To this extent the stop has been effected. However, there remains the question of the further fate of this cell. Indeed, this cell is now a **U**; hence the next stimulus [0] traveling through the cycle will convert it into an $\mathbf{S}_\theta$ (unless a stimulus [1] does this, by reaching the cell earlier via b''). After this, stimuli ([0] via the cycle of [1] via b''), as well as their absence, will transform it further, through the sensitized states {the $\mathbf{S}_\Sigma$ with Σ from list (S.1) of Sec. 2.8.1}, to an (ordinary or special) transmission state or to the confluent state {**T**, **T**′ or **C**, i.e., an $\mathbf{S}_\Sigma$ with Σ from list (S.2) of Sec. 2.8.1}. Which of these terminal states will actually be reached depends on the stimuli circulating in the cycle (plus stimuli possibly delivered via b''), i.e., on the sequence $\overline{\overline{i^1 \cdots i^n}}$, as well as on the phase of this occurrence.

In order to have a scheme that will work for all $\overline{\overline{i^1 \cdots i^n}}$ and—what is more important—for all phases in which the stop may be ordered, it is best to see to it that **U** gets an adequate and uninterrupted sequence of stimuli in any case. Hence it is best to deliver these from b''. Consequently they will be stimuli [1]. The first one will convert **U** into $\mathbf{S}_\theta$, and three more will transform this into $\mathbf{S}_{111} = \mathbf{C}_{00}$. These stimuli [1] follow the original stimulus [1], which produced the **U**, immediately. Hence a total of five consecutive

[1] [As explained in Sec. 2.8.1 above, "[0]" represents an ordinary stimulus, and "[1]" represents a special stimulus.]

stimuli [1] at b'' is called for, i.e., five consecutive stimuli [0] at a''. However, before we discuss these any further, let us consider the sequence of transformations that have been induced:

$$(4') \qquad \begin{aligned} &\text{Cycle cell (one of the } \mathbf{C}, \mathbf{X}, \overset{0}{\rightarrow}, \downarrow 0, \overset{0}{\leftarrow}) \rightarrow \\ &\rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_1 \rightarrow \mathbf{S}_{11} \rightarrow \mathbf{S}_{111} = \mathbf{C}_{00}\,. \end{aligned}$$

In the end it will be necessary to restore the cycle cell that has been so transformed to its original condition. It is simplest to use the sequence (4′) itself for this purpose. This sequence ends with a **C**; hence it should have begun with a **C**, too. That is, the cycle cell in question should be the **C** in the cycle. This means that the b'' of Figure 18g must be the b_1 of Figure 18f. Consequently the (ultimate) output b must be b_2.

Let us now return to the five stimuli [0] at a''. These could be furnished by a pulser $\mathbf{P}(\overline{11111})$, whose output b^* will then feed a'', while its input a^* is fed by a_-. However, one more point needs to be considered.

The above 5 stimuli paralyze the output of the cycle cell of (4′) at the time when it goes into the state **U**, also at the 4 successive times when it goes into the states $\mathbf{S}_\theta$, $\mathbf{S}_1$, $\mathbf{S}_{11}$, $\mathbf{S}_{111} = \mathbf{C}_{00}$; and finally, since **C** responds with a delay 2, there will also be no output from this cell at the next time. That is, this cell is silent for 6 consecutive times. If the length n of the period is $\leqq 6$, then this is sufficient to silence the cycle for good. However, if $n > 6$, then this sequence of 5 stimuli [1] at b'', i.e., of 5 stimuli [0] at a'', must be repeated. Let us repeat it p times. It is clear from the above, that we could insert a 0 (no stimulus) between every two groups of $\overline{11111}$ (5 stimuli), but it is simpler to omit this insertion. So we have a sequence $\overline{1 \cdots 1}$ of order $5p$. This will take the cycle cell p times through the sequence (4′), and thereby silence the cycle (i.e., its output $b = b_2$) for $5p + 1$ consecutive times. Hence it will silence it for good if $n \leqq 5p + 1$, i.e., if

$$(5') \qquad p \geqq \frac{n-1}{5}.$$

The simplest choice of p is, of course, as the smallest integer that satisfies condition (5′).

Thus the pulser referred to above (with its output b^* feeding a'' and its input a^* fed from a_-) must be $\mathbf{P}(\overline{1 \cdots 1})$ of order $5p$. The entire arrangement is shown in Figure 18h. Starring all quantities that refer to this pulser, we see that $n^* = k^* = 5p$ and $\nu_h{}^* = h (h =$

$1, \cdots, 5p$) obtain; hence $u^{*0} = 0$ by rule (2′) of Section 3.2.1, so that we can choose $u^* = 0$.

The border of **U**'s around the right and lower sides of the pulser in Figure 18h serves again to avoid the possibility of unwanted stimulations due to border **C**'s.

Before continuing, we will point out an important special case, where the introduction of the pulser of Figure 18h can be avoided. This is the case of the simplest possible period, namely $\bar{\bar{1}}$. This has the order $n = 1$. In this case we proceed as follows.

The order $n = 1$ violates the condition (3′); hence we must $\mathfrak{o}$-fold it, so that $\mathfrak{o}n$ fulfills (3′), i.e., with an $\mathfrak{o} \geqq 5$. (This would suggest $\mathfrak{o} = 5$, but we postpone the choice.) The new n is $\mathfrak{o}$ (the previous $\mathfrak{o}n$); the cycle in Figure 18f has the period $n = \mathfrak{o}$. Now assume that a stimulus [1] at b_1 of Figure 18f (i.e., at b'' of Fig. 18g; cf. above), converts the right upper **C** in this cycle (Fig. 18f) into a **U**. This starts the sequence (4′). To complete this sequence 4 more stimuli are needed. They are rigidly timed. In our discussion after sequence (4′) we made them [1]'s from b'', but [0]'s from the cycle would do just as well. After the cycle cell has undergone its first transformation in (4′) {i.e., (**C** → **U**)}, the cycle will still deliver $n - 2$ stimuli [0] to it. If $n - 2 = 4$, i.e., $n = 6$, then this is precisely what is needed to complete the sequence (4′). At the same time the cycle will have been silenced for good, as desired (cf. the discussion leading to condition (5′), or condition (5′) itself with $n = 6$, $p = 1$). Hence we choose $n = \mathfrak{o} = 6$. (Note that this fulfills our earlier condition $\mathfrak{o} \geqq 5$, but it is not the minimum choice $\mathfrak{o} = 5$ referred to there!)

With this choice, then, a single stimulus [1] at b'' (Fig. 18g) suffices. In this case, therefore, the addition of Figure 18h to Figure 18g is not needed. However, it is more convenient in this case to have the input a'' on the left, rather than on the upper side of the **C**. Also, since the pulser $\mathbf{P}(\overline{1 \cdots 1})$ of Figure 18h is now omitted, it is convenient, for the unification of the notations, to identify a'' and a^*. This is shown in Figure 18i.

We can now perform the main synthesis by joining the network of Figure 18f with the network of Figure 18h {the general case: $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$} or the network of Figure 18i {the special case: $\mathbf{PP}(\bar{\bar{1}})$}. As noted before, the contact must be made at $b'' = b_1$, and the (ultimate) output is $b = b_2$. The result is shown in Figures 18j and 18k, respectively. Note that in the second case $n = 6$, also $h = 6$, $\nu_h = h (h = 1, \cdots, 6)$; hence by formula (2′) we have $u^0 = 0$, so that we can choose $u = 0$. Consequently, $2k = 12$ and $u + 2 = 2$,

as shown in Figure 18k. Note also that though Figure 18j is drawn as if $\ell < u + 2$ and $10p < 2k + 1$, either or both of $\ell \geqq u + 2$ and $10p \geqq 2k + 1$ are equally possible.

Figures 18j and 18k contain certain delays, from the stimulation at a_+, or rather at a', to the start at b, and from the stimulation at a_-, or rather at a^*, to the stop at b. We will determine these.

Consider the first case, i.e., Figure 18j.

The path from a' to $b_2 = b$ in Figure 18j lies entirely in the lower half of the network, i.e., in the part shown in Figure 18f. Using Figure 18f, we saw at the end of Section 3.2.1 that the stimuli of $\overline{i^1 \cdots i^n}$ appear at b' with an absolute delay of $2k + u + 1$ after the stimulus at a'. To this absolute delay must be added the relative delays $1, \cdots, n$, so that the sequence $\overline{i^1 \cdots i^n}$ appears at b' with the delays $2k + u + 2, \cdots, 2k + u + n + 1$. The delay from b' to b is clearly 4; hence the first period of $\overline{\overline{i^1 \cdots i^n}}$ appears at b with the delays $2k + u + 6, \cdots, 2k + u + n + 5$. Thus the start at b ($b = b_2$; cf. Fig. 18j) is delayed against the stimulus at a' by $2k + u + 6$.

The path from a^* to $b_2 = b$ in Figure 18j consists of the part from a^* to $b_1 = b''$, followed by the part from $b_1 = b''$ to $b_2 = b$. The part from a^* to b'' lies entirely in the upper half of the network, i.e., in the part shown in Figure 18h. Using Figure 18h, we saw at the end of Section 3.2.1 that the stimuli $\overline{1 \cdots 1}$ (order $5p$) appear at b^* with an absolute delay of $2k^* + u^* + 1$ after the stimulus at a. Since $k^* = 5p$, $u^* = 0$, this delay is $10p + 1$. To this absolute delay must be added the relative delays $1, \cdots, 5p$, so that the sequence $\overline{1 \cdots 1}$ appears at b^* with the delays $10p + 2, \cdots, 15p + 1$. The delay from b^* to b'' is clearly 5; hence the sequence $\overline{1 \cdots 1}$ appears at $b_1 = b''$ with the delays $10p + 7, \cdots, 15p + 6$. Now a stimulus [1] at b_1 (Fig. 18f) inhibits the next output from the affected **C** {cf. the discussion of sequence $(4')$}. Hence the output at b_2 is stopped with a total delay of $10p + 8$. Thus the stop at b ($b = b_2$, cf. Fig. 18j) is delayed against the stimulus at a^* by $10p + 8$.

Consider now the second case, i.e., Figure 18k.

The lower halves of Figures 18j and 18k have the same structure; hence the delay from a' to the start at b is given by the formula derived above for the first case. This is $2k + u + 6$. Since now $k = 6$, $u = 0$ (cf. the discussion of Fig. 18k), this delay is 18.

The delay from $a^* = a''$ to the stop at $b_2 = b$ in Figure 18k is also easy to determine. The path from a'' to $b_1 = b''$ is also shown in Figure 18i. Its delay is clearly 3. The delay from a stimulus [1] at

b_1 to a stop at b_2 (Fig. 18f) is the same as in the first case, that is, 1. Thus the stop at b ($b = b_2$; cf. Fig. 18k) is delayed against the stimulus at a^* ($= a''$; cf. Fig. 18k) by 4.

We restate: The delay from a stimulus at a' to the start at b is $2k + u + 6$ in the first (general) case (Fig. 18j) and 18 in the second (special) case (Fig. 18k). The delay from a stimulus at a^* to the stop at b is $10p + 8$ in the first (general) case (Fig. 18j) and 4 in the second (special) case (Fig. 18k).

In contrast with the situation at the end of Section 3.2.1, corruption by interference in this network is a definite possibility. It is easily verified that this is controlled by the following rules.

(6′.a) Throughout (6′.b)–(6′.d) stimulations at a', a^* should be viewed in a modified chronological ordering, which differs from the ordinary one by certain systematic (relative) shifts. All statements in (6′.b)–(6′.d) about simultaneity, precedence, and ordering of such stimuli must accordingly be understood in this modified chronology. This chronology is defined as follows. The ordering of stimulations at a' relatively to each other is unchanged; also the ordering of stimulations at a^* relatively to each other is unchanged. Stimulations at a' are displaced relatively to stimulations at a^* in accordance with the difference between the delays from a' to the start at b, and from a^* to the stop at b. In other words, all these stimulations are ordered chronologically, not by the times of their respective occurrences (at a' or at a^*), but by the times at which they take effect (at b, in the form of a start at b for a', and of a stop at b for a^*).

(6′.b) Multiple stimulations at a', between which no stimulations at a^* occur, simply superposes their effects. One could therefore say, as at the end of Section 3.2.1, that no corruption by interference takes place in this case. However, it must be noted that this means that the periodic emission of the organ (at b) may be changed in this process, since each new stimulation at a' superposes the period with its own phase on the (possibly already composite) period produced by the previous stimulations. Finally, this change is void in the special case, where the period consists of stimuli only (it is $\overline{\overline{1}}$!), so that additional superpositions cannot alter it.

(6′.c) A stimulation at a^* which follows upon stimulations at a' {for these cf. rule (6′.b)} stops the period generated by these. Multiple stimulations at a^* have no effect beyond that one of the first one among them; i.e., the stop caused by the first one is maintained by the others, but it would be equally maintained without them.[2] It should be noted that if stimulations at a' and at a^* are simultaneous, the latter overrides the former; i.e., no emissions at b will occur.

(6′.d) A stimulation at a' which follows upon stimulations at a^* takes full effect {in the sense of (6′.b)} only if it has a delay $\geqq 5p + 1$ (this is the general case; in the special case put $p = 1$) against the former. If it comes earlier, then those stimuli in the period whose first occurrences would take place with a delay $\leqq 5p + 1$ will be perpetually removed from the period.

To conclude, we will make some adjustments to equalize the delays from a' to the start at b and from a^* to the stop at b, thereby eliminating the complications caused by the special ordering of (6′.a) in (6′.b)–(6′.d). This will be done by inserting suitable delay paths between a_+ and a' on one hand, and between a_- and a^* on the other. We will distinguish three sub-cases, as follows.

First, assume the case of Figure 18j and $10p < 2k + 1$. Define z_1 by $10p + z_1 = 2k + 1$, so that $z_1 = 1, 2, \cdots$. Attach the network shown in Figure 19a to the left side of Figure 18j. The delay from a_+ to a' is 3; hence the delay from a_+ to the start at b is $3 + (2k + u + 6) = 2k + u + 9$. The delay from a_- to a^* is $u + z_1$; hence the delay from a_- to the stop at b is $(u + z_1) + (10p + 8) = (10p + z_1) + (u + 8) = (2k + 1) + (u + 8) = 2k + u + 9$.

Thus both delays are $2k + u + 9$, and Figures 18j and 19a put together assume the aspect of Figure 19c. This figure is drawn as if $\ell < u + 2$, but $\ell \geqq u + 2$ is equally possible.

Second, assume the case of Figure 18j and $10p \geqq 2k + 1$. Define z_2 by $(2k + 1) + z_2 = 10p$, so that $z_2 = 0, 1, 2, \cdots$. Attach the network shown in Figure 19b to the left side of Figure 18j. The delay from a_+ to a' is $3 + z_2$; hence the delay from a_+ to the start at b is $(3 + z_2) + (2k + u + 6) = ((2k + 1) + z_2) + (u + 8) = 10p + u + 8$. The delay from a_- to a^* is u; hence the delay from a_- to the stop at b is $u + (10p + 8) = 10p + u + 8$.

Thus both delays are $10p + u + 8$, and Figures 18j and 19b put

[2] [There is an error here, which we will discuss at the end of the subsection.]

together assume the aspect of Figure 19d. This figure is drawn as if $\ell < u + 2$, but $\ell \geqq u + 2$ is equally possible.

Before going on to the third case, note that the common delay in the first case (where $10p < 2k + 1$) is $2k + u + 9$, while in the second case (where $10p \geqq 2k + 1$) it is $10p + u + 8$. Both cases are covered by the expression $M + u + 8$, where

$$M = \text{Max } (10p, 2k + 1). \tag{7'}$$

Also, Figures 19c and 19d have in every way the same outward appearance, except that the first has the width $2k + 4$, while the second has the width $10p + 3$. Again both cases are covered by one expression, namely by $M + 3$. Hence Figure 19e represents both Figures 19c and 19d. Figure 19e is drawn as if $\ell < u + 2$, but $\ell \geqq u + 2$ is equally possible.

Third, assume the case of Figure 18k. Attach the structure shown in Figure 19f to the left side of Figure 18k. Also, identify a' with a_+. The delay from a_+ to the start at b is then the same as that one from a', that is, 18. The delay from a_- to a^* is 14; hence the delay from a_- to the stop at b is $14 + 4 = 18$.

Thus both delays are 18, and Figures 18k and 19f put together assume the aspect of Figure 19g.

Figures 19d and 19g are the desired network in the case of a general $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$ and of the special $\mathbf{PP}(\bar{\bar{1}})$ {represented with $n = 6$, i.e., as $\mathbf{PP}(\overline{\overline{111111}})$}, respectively. It is worthwhile to introduce a joint abbreviated representation for these. In the case of Figure 19d we bring it to the uniform height of $N + 3$, where

$$N = \text{Max } (u + 2, \ell), \tag{8'}$$

by filling in **U**'s as needed. Figure 19g need not be changed at all. The result is shown in Figure 19h. For the general case of $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$:

$$K = M + 3,\ L = N + 3,\ L_1 = 2,\ L_2 = 4,\ L_3 = L - 4.$$

For the special case of $\mathbf{PP}(\bar{\bar{1}})$:

$$K = 15,\ L = 4,\ L_1 = 0,\ L_2 = 3,\ L_3 = 1.$$

The delays from a_+ to the start at b and from a_- to the stop at b are the same. This common delay is $M + u + 8$ in the general case and 18 in the special case. We can now reformulate the rules (6.a)–(6.d), with a_+ in place of a' and a_- in place of a^*, and with the common delay for starting and stopping. This eliminates, as observed earlier, the need for the special ordering of rule (6'.a). We obtain accordingly this.

The rules that govern possible corruption by interference are now modified from their form in (6′.a)–(6′.d) in this way:

(9′) The special chronology for stimulations at a' and a^*, as defined in (6′.a), is now replaced by the ordinary chronology of the occurrence of stimulations at a_+ and a_-. That is, all comparisons must now be made according to the times at which stimulations occur at a_+ and a_-. With this modification (and replacing a' by a_+ and a^* by a_-) the rules of (6′.b)–(6′.d) remain valid.

[As noted above, there is an error in von Neumann's rule (6′.c) concerning the timing of a periodic pulser. There is also an important case which von Neumann didn't discuss, namely, the case in which a "stop" stimulus precedes a "start" stimulus.

Let us consider first the timing of von Neumann's periodic pulsers, using Figure 17 as an example. For the periodic pulser **PP** ($\overline{10010001}$) we have the following phasing. A "start" stimulus into input a_+ at time t will cause the sequence $\overline{10010001}$ to enter cell *H4* from cell *G4* at times $t + 31$ through $t + 38$. A "stop" stimulus into input a_- at time t will cause the sequence $\overline{1111111111}$ to enter cell *H4* from cell *H3* at times $t + 32$ through $t + 41$. In this case nothing is emitted from the output b, since the killing signal from *H3* dominates the transmission signal from *G4*. If the "stop" stimulus comes one moment later (at time $t + 1$), the sequence $\overline{1111111111}$ enters cell *H4* from cell *H3* at times $t + 33$ through $t + 42$. In this case cell *H4* is in state $\mathbf{C}_{00}$ at time $t + 31$, in state $\mathbf{C}_{01}$ at time $t + 32$, in state $\mathbf{C}_{10}$ at time $t + 33$ (and a pulse is emitted from b at time $t + 33$), and in state **U** at time $t + 34$.

Consider next the phasing of von Neumann's special periodic pulser **PP** ($\overline{1}$). A "start" stimulus into input a_+ at time t produces the sequence $\overline{111111}$ into cell *E3* from cell *D3* at times $t + 16$ through $t + 21$. A "stop" stimulus into input a_- at time t enters cell *E3* from cell *E2* at time $t + 17$. Consequently, nothing is emitted from b if the "start" and "stop" stimuli are simultaneous. If the "stop" stimulus comes one moment later, then one pulse is emitted from b, as was the case with the periodic pulser **PP** ($\overline{10010001}$).

It should be noted that we have measured the delay from the input a_+ and a_- up to (but not including) the cell containing the confluent state **C** which drives the output b. Von Neumann measured these delays through this cell to the output b. For a signal from the "stop" input a_- the delay through this final cell is only 1 unit, since killing takes only 1 unit of time. Hence the delay from both a_+ and a_- to

the stop at b is 33 for $\mathbf{PP}(\overline{\overline{10010001}})$ and 18 for von Neumann's $\mathbf{PP}(\overline{\overline{1}})$. See Table I.

Thus the phasing for all periodic pulsers $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$ constructed by von Neumann's algorithm is as follows. Assume the periodic pulser is as designed and is quiescent. If the "start" stimulus into input a_+ and the "stop" stimulus into input a_- are simultaneous, nothing is emitted from output b, and the periodic pulser is left in its initial state. If the "stop" stimulus follows the "start" stimulus by $T (T > 0)$ units of time, the sequence $\overline{i^1 \cdots i^n}$ is emitted ν times and then an initial sequent of it of length μ is emitted, where $T = n\nu + \mu$, $\mu < n$, and either ν or μ may be zero. This phasing is as von Neumann intended and is in accord with his rules. But von Neumann did not consider all the cases in which the "stop" stimulus precedes the start stimulus (i.e., $T < 0$). Apparently, he planned to use each periodic pulser in such a way that there would be a one to one correspondence of "start" to "stop" stimuli, with the m'th "stop" stimulus coming no earlier than the m'th "start" stimulus, with the m'th "stop" stimulus preceding the $(m + 1)$st "start" stimulus by a sufficient length of time for the periodic repeater to be cleared properly. But he did not state this intention, and in his subsequent use of periodic pulsers he did not always conform to it.

Consider the effect of feeding the sequence $\overline{101}$ into the "stop" input a_- of $\mathbf{PP}(\overline{\overline{10010001}})$. This sequence will cause the upper pulser $\mathbf{P}(\overline{1111111111})$ to emit 12 (rather than 10) pulses and (depending on the phasing of the "start" input a_+) may leave cell $H4$ in an undesired state. For some conditions of the phasing of $\overline{101}$ into a_-

TABLE I

External characteristics of periodic pulsers

	General Case		Special Case	
	In terms of parameters	Example of $\mathbf{PP}(\overline{\overline{10010001}})$	Von Neumann's $\mathbf{PP}(\overline{\overline{1}})$	Alternate $\mathbf{PP}(\overline{\overline{1}})$
Width K	$M + 3$	23	15	13
Height L	$N + 3$	10	4	4
Number of cells below an input or output				
Input a_+	2	2	0	1
Input a_-	4	4	3	3
Output b	$L - 4$	6	1	1
Delay from input a_+ to b. Also, delay from input a_- to stop at b.	$M + u + 8$	33	18	19

and $\overline{1}$ into a_+, the cell *H5* may be killed to **U**. This difficulty may be solved by the following *rule of usage*. The "stop" input a_- of a "general case" periodic pulser $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$ is never to be stimulated twice in any time span of length n. Actually, all of von Neumann's later uses of "general case" periodic pulsers conform to this rule.

There is a more serious problem in von Neumann's "special case" periodic pulser $\mathbf{PP}(\overline{\overline{1}})$. A stimulus into the "stop" input a_- changes cell *E3* into **U**, and then the four pulses in the pulser $\mathbf{P}(\overline{111111})$ or the periodic repeater (or both) operate by the direct process to leave *E3* in the confluent state **C**. But suppose a "stop" stimulus precedes a "start" stimulus. Then cell *E3* will not be left in the correct state! This fact causes no trouble when von Neumann's $\mathbf{PP}(\overline{\overline{1}})$ is used in the triple-return counter (Sec. 3.4 below), but it does cause incorrect operation of his control organ **CO** as it is used in the read-write-erase-control **RWEC** (Sec. 4.3.4 below). We could modify some of the contextual circuitry of the **CO**, but it is better to use a $\mathbf{PP}(\overline{\overline{1}})$ in the **CO** which is not contaminated if it is turned off before being turned on.

This alternate $\mathbf{PP}(\overline{\overline{1}})$ is shown in Figure 20. It uses two modified pulsers $\mathbf{P}(\overline{11111})$, one above and one below. A "start" stimulus into input a_+ at time t causes the sequence $\overline{11111}$ to enter cell *G3* from cell *G4* at times $t + 17$ through $t + 21$. If cell *G3* is in the confluent state **C** this sequence will emerge from the output b at times $t + 19$ through $t + 23$, so the delay from input a_+ to b is 19 units of time. A "stop" stimulus into input a_- at time t causes the sequence $\overline{11111}$ to enter cell *G3* from cell *G2* at times $t + 18$ through $t + 22$. Since killing takes only 1 unit of time, the delay from input a to the stop at b is 19 units, which is the way von Neumann counted it. See Table I.

Note further that if the start input a_+ is stimulated from 1 to 4 units of time after the stop input a_- is stimulated, from 1 to 4 pulses will be left in the repeater. Hence we stipulate the following rule of usage for the alternate $\mathbf{PP}(\overline{\overline{1}})$ of Figure 20: the stop input a_- is never to be stimulated twice within 5 units of time, and the start input a_+ is never to be stimulated from 1 to 4 units of time after the stop input is stimulated. The structure of the read-write-erase-control unit **RWEC** is such that this rule is always satisfied. When operated in an environment which conforms to this rule, the alternate $\mathbf{PP}(\overline{\overline{1}})$ functions as a "flip-flop." It is turned on at a_+, off at a_-, and while it is on it emits a continuous sequence of stimuli from b which may be used to operate a gate (confluent state).

We will now summarize the external characteristics of periodic pulsers. Von Neumann's special case $\mathbf{PP}(\overline{\overline{1}})$ is shown in Figure 17b.

Our alternate special case **PP**$(\overline{\overline{1}})$ is shown in Figure 20. The external characteristics of both of these **PP**$(\overline{\overline{1}})$ are given in Table I.

The general case **PP**$(\overline{\overline{i^1 \cdots i^n}})$, with $n \geqq 5$, is shown in Figure 19h. If $n < 5$ and the special case does not apply, the characteristic is iterated until $n \geqq 5$; for example, if **PP**$(\overline{101})$ is asked for, **PP**$(\overline{\overline{101101}})$ is constructed. The general case **PP**$(\overline{\overline{i^1 \cdots i^n}})$ contains the pulser **P**$(\overline{i^1 \cdots i^n})$, for which u and k are defined at the end of Section 3.2.1. The following string of definitions leads to the parameters K (width) and L (height) of **PP**$(\overline{\overline{i^1 \cdots i^n}})$.

$$\begin{aligned} \ell &= \text{integer part of } (n-1)/2 \\ N &= \text{Maximum of } u+2 \text{ and } \ell \\ L &= N+3 \\ p &= \text{The smallest integer such that } 5p \geqq n-1 \\ M &= \text{Maximum of } 10p \text{ and } 2k+1 \\ K &= M+3. \end{aligned}$$

Further information is given in Table I.]

3.3 The Decoding Organ: Structure, Dimensions, and Timing

[Figure 21 shows the decoding organ **D**$(\overline{10010001})$. This decoding organ has characteristic $\overline{10010001}$ and order 8. All sequences of length 8 can be divided into two classes: those which are bitwise implied by (contain all the stimuli of) the characteristic $\overline{10010001}$ (e.g., $\overline{10010001}$ and $\overline{11010011}$), and those which are not (e.g., $\overline{10000001}$ and $\overline{10010010}$). If a sequence of the former class is fed into input a, a single stimulus will come from output b after a suitable delay, while a sequence of the latter class produces no output.

A decoding organ is to be distinguished from a "recognizing" device which produces an output for some particular sequence (e.g., $\overline{10010001}$) and no other. Such devices are discussed in Section 3.5.

The decoder **D**$(\overline{10010001})$ works in this manner. Suppose the sequence $\overline{i^1 i^2 i^3 i^4 i^5 i^6 i^7 i^8}$, with $i^1 = 1$, $i^4 = 1$, and $i^8 = 1$, enters input a at times t through $t+7$; the three stimuli i^1, i^4, and i^8 enter with relative delays 0, 3, and 7, respectively. Paths B and D meet at the confluent state $D1$ with delays of 21 and 18, respectively; hence i^1 and i^4 will produce an output from cell $D1$ at time 23. This output will arrive at cell $F1$ at time 24, coincident with the arrival of i^8 via path F, thereby producing an output from b at time 26. Note that since there are three ones in the characteristic $\overline{10010001}$, three paths (B, D, and F) are needed in the decoder.

The design of a decoding organ is very similar to that of a pulser (Sec. 3.2.1). Indeed, a pulser is in fact a coder. For the decoding

organ, however, confluent states are needed along the top row to detect coincidence. When confluent states are needed to introduce relative delays between paths of odd amounts (e.g., as in cell *B3* of Fig. 21), then these confluent states must be isolated from those in the top row by means of $\mathbf{T}_{01\epsilon}$ states (as in row 2 of Fig. 21).]

Our third construction is the *decoding organ*. This organ has an input a and an output b. The ideal form would be one which, upon the arrival of a prescribed sequence, say $\overline{i^1 \cdots i^n}$, at a, and only then, will emit a single stimulus at b. However, for our specific applications of this organ a simpler requirement will do, and this is the one that we are going to use. It is as follows. Let a sequence, e.g., $\overline{i^1 \cdots i^n}$, be prescribed. Upon arrival of any sequence $\overline{j^1 \cdots j^n}$ at a, which contains all the stimuli of $\overline{i^1 \cdots i^n}$ (i.e., such that $i^\nu = 1$ implies $j^\nu = 1$), and only then, it will emit a single stimulus at b.

The relation between the actuating sequence $\overline{j^1 \cdots j^n}$ at a and the response at b is freely timed; i.e., the delay between these is not prescribed at this point (cf., however, the remarks at the end of this section).

This decoding organ has the symbol $\mathbf{D}(\overline{i^1 \cdots i^n})$. The sequence $\overline{i^1 \cdots i^n}$, which can be prescribed at will, is its *characteristic*, and n is its *order*.

The required network is successively developed in Figure 22. This construction and the network that results from it are discussed in the balance of this section. They have a great deal in common with those for the pulser $\mathbf{P}(\overline{i^1 \cdots i^n})$, as discussed in Section 3.2.1.

Let $\nu_1, \cdots, \nu_k$ be the ν for which $i^\nu = 1$, i.e., the positions of the stimuli in the sequence $\overline{i^1 \cdots i^n}$. (At this occasion, unlike in Sec. 3.2.1, we are not interested in their monotone ordering.) Write

$$(10') \qquad \begin{cases} n - \nu_h = 2\mu_h' + r_h', \\ \text{where } \mu_h' = 0, 1, 2, \cdots; r_h' = 0, 1. \end{cases}$$

{Note the difference between equation (10′) and equation (1′)!}

The input stimuli at a must be compared, to see whether k stimuli with the relative delays $\nu_1, \cdots, \nu_k$ are present there. This can be done by multiplexing each stimulus that arrives at a to k distinct stimuli, numbers $h = 1, \cdots, k$, arriving at some comparison point b' with the relative delays $n - \nu_h$ $(h = 1, \cdots, k)$, respectively. Then the simultaneous arrival of k stimuli at b' is equivalent to the arrival of k distinct stimuli, numbers $h = 1, \cdots, k$, arriving at a with the relative delays $\nu_1, \cdots, \nu_k$.

Hence we need k paths, numbers $h = 1, \cdots, k$, from a to b', which have the relative delays $n - \nu_h$ $(h = 1, \cdots, k)$, respectively, with

respect to each other. However, we cannot sense a k-fold coincidence by a single act, at a single point, if $k > 3$. ($k = 3$ is the maximum that a single **C** can handle; cf. Sec. 2.3.2.) It is therefore best to bring these k paths together pairwise. First, paths 1 and 2 merge at a comparison point b_2' (it is better to begin with b_2' rather than with b_1'; cf. below), so that joint path $2'$ continues from b_2'; then paths $2'$ and 3 merge at a comparison point b_3', so that joint path $3'$ continues from b_3'; then paths $3'$ and 4 merge at a comparison point b_4', so that joint path $4'$ continues from b_4'; $\cdots$; and finally paths $(k-1)'$ and k merge at b_k', so that joint path k' continues from b_k' directly to the output b. In this successive way 2-fold coincidences at b_2', $\cdots$, b_k replace the single k-fold coincidence at b'. A **C** can, of course, handle a 2-fold coincidence without difficulty (cf. Sec. 2.3.2).

This procedure clearly calls for a network of the type developed in Figures 16a and 16b. Consider first Figure 16a. Here the stimulus entering at a reaches b over k different paths, and while the delays that are desired for each one of these paths are not yet properly adjusted, this can be attended to afterwards, with the means of Figure 16b, i.e., with the networks N of Figure 16c, as detailed in Figures 16d–16f. We discuss first another imperfection that Figure 16a presents in the present situation. This is the following.

In Figure 16a the k paths that are produced by the **C** on the base line are directly merged by those $\underset{\rightarrow}{0}$ on the top line that lie above the **C** (on the base line) numbered 2, 3, $\cdots$, k. These mergers are effected by $\underset{\rightarrow}{0}$ cells, i.e., without any coincidence requirements. This is conformal to the purposes of Figure 16a in Section 3.2.1 but not to our present ones. The $k-1$ cells $\underset{\rightarrow}{0}$ in question, where these mergers occur, are obviously the comparison points b_2', b_3', $\cdots$, b_k', referred to above. Hence these must be able to sense coincidences; i.e., they should be **C**, not $\underset{\rightarrow}{0}$. For reasons of symmetry we also replace the $\underset{\rightarrow}{0}$ on the top line above the **C** (on the base line) numbered 1, by **C**, and call it b_1'. This new arrangement is shown in Figure 22a.

A stimulus arriving at a has k paths available to reach b. On path h ($h = 1, \cdots, k$) it goes horizontally from a to the (base line) **C** numbered h; then it turns vertically up, ascends to the top line (to the **C** at b_h') and continues there horizontally to b. (Up to this point, and not further, we are following the pattern of Fig. 16a.) These are $(2h-1)+u+(2(k-h)+1) = 2k+u$ steps, but since $h+(k-h+1) = k+1$ of them are **C**'s, the entire delay involved is $3k+u+1$. That is, the delay is the same on all paths.

In view of this we must increase the delay on path number h by $n - \nu_h = 2\mu_h' + r_h'$. {Cf. equation $(10')$.} This can be achieved

exactly as was its analog in Section 3.2.1: each vertical branch is replaced by a suitable delay network, say branch number h by a network N_h'. This N_h' is like the N_h of Figure 16e; that is, it is made up of the parts shown in Figures 16d–16f, except that now μ_h, r_h are replaced by μ_h', r_h'. Thus Figure 22b results, which is related to Figure 22a in the same way that Figure 16b is related to Figure 16a. The height u obtains by repetition of the considerations of Section 3.2.1 that led up to condition $(2')$. {Now μ_h, r_h are replaced by μ_h', r_h', i.e., the $\nu_h - h$ of $(1')$ are replaced by the $n - \nu_h$ of equation $(10')$.} We obtain the condition $u \geqq u'^0$ (in analogy to condition $(2')$, with the modifications mentioned above), where

$$(11') \quad \begin{cases} u'^0 = \text{Max } (n - \nu_h) + \epsilon'^0, \\ \text{where } \epsilon'^0 = 1 \text{ if the Max is even but some } n - \nu_h \text{ is odd,} \\ \text{and } \epsilon'^0 = 0 \text{ otherwise.} \end{cases}$$

It is, of course, simplest to put $u = u'^0$.

[Von Neumann overlooked the need for an extra row of cells between row u of Figure 22a and the top row in case row u contains a confluent state. Otherwise, the confluent state of row u will be adjacent to a confluent state in the top row, creating an open path, since a confluent state cannot directly drive another confluent state. Row 2 of Figure 21 is this extra row which von Neumann overlooked; if row 2 were deleted, the confluent states of cells *B1* and *B3* would be adjacent.

This oversight may be remedied without changing von Neumann's parameters as follows. Let $1 \leqq \nu_1 < \nu_2 < \cdots < \nu_k \leqq n$. Then there are two cases, according to whether an extra confluent state occurs in the leftmost path (i.e., the path for ν_1) or another path. If an extra confluent state occurs in the leftmost path, replace it and the confluent state above it (i.e., in the top row) by a block of four connected ordinary transmission states. Thus in Figure 21, put a block of four ordinary transmission states in cells *A1*, *B1*, *A3* and *B3* and delete row 2. If an extra confluent state occurs in a path other than the leftmost path, place it one cell lower.]

This completes the construction. As Figure 22b shows, the area of this network has the width $2k$ and the height $u + 2$. An abbreviated representation of this network is given in Figure 22c.

The delay from a to b through Figure 22a was $3k + u + 1$ on each path. Through Figure 22b it is therefore $1 + (3k + u + 1) + (n - \nu_h)$ on path number h. The first term, 1, is due to the insertion of the $\underset{\rightarrow}{0}$ before the first **C** in Figure 22a. Hence stimulus number ν_h (with

$i^{\nu_h} = 1$) in the sequence $\overline{i^1 \cdots i^n}$ at a reaches b with a delay $(3k + u + 2) + n$. This is counted from the time immediately before the start of the sequence $\overline{i^1 \cdots i^n}$; hence the delay counted exactly from its start is $3k + u + n + 1$. This is the same for all $h = 1, \cdots, k$, as it should be. Hence the stimulus indicating this k-fold coincidence (i.e., the presence of a sequence $\overline{j^1 \cdots j^n}$, containing $\overline{i^1 \cdots i^n}$, at a; cf. above) will appear at b with a delay $3k + u + n + 1$. Thus, in the final arrangement, represented by Figure 22c, there is from the arrival of a sequence $\overline{j^1 \cdots j^n}$ (containing $\overline{i^1 \cdots i^n}$) a delay $3k + u + n + 1$ to the response at b.

It is easily seen that in this network no corruption by interference occurs. That is, whatever stimuli may arrive at a, whenever there occurs a sequence $\overline{j^1 \cdots j^n}$ among them that contains $\overline{i^1 \cdots i^n}$, there will be a response stimulus at b, with a delay $3k + u + n$ from the beginning of that sequence.

[We will summarize the external characteristics of the decoding organ $\mathbf{D}(\overline{i^1 \cdots i^n})$. See Figure 22c.

The width of the pulser is $2k$, where k is the number of 1's in the characteristic $\overline{i^1 \cdots i^n}$. Von Neumann implicitly assumed that $k \geqq 2$; for $k = 0$ and $k = 1$ no organ is needed.

The height of the decoding organ is $u + 2$, where u is defined as follows. The $\nu_1, \cdots, \nu_k$ are the ν for which $i^\nu = 1$. Note that n is the length of the characteristic and the ν_1 is the superscript of the first 1; hence $n - \nu_1$ is the number of bits in the characteristic which are to the right of the first 1. Since all ν_h $(h = 1, \cdots, k)$ are positive, Max $(n - \nu_h) = n - \nu_1$. Von Neumann's rule for u then becomes

$u = (n - \nu_1) + \epsilon'^0$, where

$\epsilon'^0 = 1$ if $(n - \nu_1)$ is even but some $n - \nu_h$ $(h = 2, \cdots, k)$ is odd,

$\epsilon'^0 = 0$ otherwise.

The delay between the input signal i^1 entering input a and the output pulse leaving b is $3k + u + n + 1$.]

3.4 The Triple-return Counter

[In the preceding sections von Neumann gave algorithms for designing an arbitrary pulser $\mathbf{P}(\overline{i^1 \cdots i^n})$, an arbitrary periodic pulser $\mathbf{PP}(\overline{i^1 \cdots i^n})$, and an arbitrary decoder $\mathbf{D}(\overline{i^1 \cdots i^n})$. He next designed a specific organ, the triple-return counter Φ. The completed organ is shown in Figure 23, though not to scale. The periodic pulsers $\mathbf{PP}(\overline{1})$ are those of Figure 17, with width 15 and height 4; hence the actual width of Φ is 24 and its actual height is 26. The long lines with

arrows in Figure 23 symbolize sequences of ordinary transmission states being used for transmission only, not for disjunction.

Von Neumann needed the triple-return counter for a specific purpose, that of sending a pulse around a connecting loop ($\mathbf{C}_1$ or $\mathbf{C}_2$) of the external tape **L** three times; see Figure 37. Suppose that the secondary output d of Φ is connected to the input v_2 of connecting loop $\mathbf{C}_2$ and that the output w_2 of loop $\mathbf{C}_2$ is connected to the secondary input c of Φ. A pulse into the primary input a of Φ will go around loop $\mathbf{C}_2$ three times and will then be emitted from the primary output b of Φ.

We assume as a rule of usage that once input a of Φ is stimulated, it will not be stimulated again until a stimulus has been emitted from output b of Φ. Under this assumption, the triple-return counter Φ works like this. A start stimulus into a goes via a coding and decoding network to a_+^1, where it starts the first periodic pulser, and also to output d, where it is relayed to the input of $\mathbf{C}_2$. After a delay, the output b^1 of the first periodic pulser will send stimuli along row *14* to impinge on confluent cell *F14*, which functions as a gate. When the output pulse from $\mathbf{C}_2$ enters Φ at c, it travels along row *20* and into column D, stimulating the three gates *F14*, *F8*, and *F2*. Only the first of these gates (*F14*) is open; the pulse passes through this gate, turns the first periodic pulser off, turns the second periodic pulser on, and passes along row *13* and column J to the secondary output d.

The next pulse entering secondary input c from $\mathbf{C}_2$ turns off the second pulser, turns on the third pulser, and goes from secondary output d back to $\mathbf{C}_2$. When this pulse returns from $\mathbf{C}_2$, it turns off the third pulser and passes out the primary output b. This completes the operation of Φ.

Note that the path from primary input a to cell *F18* crosses the path from secondary input c to cell *D14*. Actually, no harm results from the primary input stimulus going to cell *D14* and thence to gates *F14*, *F8*, and *F2*, since these gates are initially closed. But if the pulses from c to *D14* entered cell *F18*, a malfunction would result. This is a special case of the general problem of wire-crossing. The general problem is solved by means of the coded channel of Section 3.6 below. The special problem is solved in Φ by means of the coder *B17*, *B18*, *C17*, *C18*, which produces $\overline{101}$ when stimulated, and the decoder *D18*, *D19*, *E18*, *E19*, *F18*, *F19*, which emits a $\overline{1}$ to *F17* and *G18* when it is stimulated by $\overline{101}$, but not when it is stimulated by $\overline{1}$.

The design principle of the triple-return counter can easily be modified and generalized to give a counter which will count m pulses; that is, will emit every m'th pulse it receives.]

The three organs that we constructed in Sections 3.2 and 3.3 were basic entities of rather general significance. The next ones will be much more special; they correspond to specific needs that will arise in the course of the first major, composite construction that we must undertake. Their names, which refer to very special functions, also express this fact.

The first organ in this series is the *triple-return counter*. This organ has two inputs a, c and two outputs b, d; a, b are the *primary* input-output pair and c, d are the *secondary* input-output pair.

In order to describe its functioning, it is necessary to assume that its secondary output d and its secondary input c are attached to the input c^* and the output d^*, respectively, of an (arbitrarily given) other organ. This other organ is the *responding* organ, and we will, for the time being, give it the symbol Ω. Having been thus attached to Ω, the triple-return organ now has only its primary input-output pair a, b free. Its desired functioning is as follows.

Upon stimulation at a it responds at d and thereby stimulates at c^*. Assume that Ω responds, after a suitable delay, at d^*, stimulating c. This causes a second response at d, and hence a stimulation of Ω at c^*. Assume that Ω responds for the second time, after a suitable delay, at d^*, stimulating at c. This causes a third response at d, and hence a stimulation of Ω at c^*. Assume that Ω responds at d^* for the third time, after a suitable delay, stimulating at c. This causes a response at b and terminates the process.

The relation between the original actuation at a and the ultimate response at b is freely timed; i.e., the delay between these is not prescribed at this point. Note that the total process from a to b has three phases, namely the three passages from d through c^* and d^* (i.e., through Ω) to c, whose delays depend in any event on the responding organ Ω, and not on the organ to be constructed now. However, the other phases of the total process from a to b (i.e., those from a to d, twice from c to d, and finally from c to b) depend solely on the organ to be constructed now. It is to these that the above observation about free timing, i.e., the absence of prescriptions of specific delay lengths, applies (cf., however, the remarks at the end of this section).

The triple-return counter has the symbol Φ.

The required network is successively developed in Figure 24. This construction and the network that results from it are discussed in the balance of this section.

We require that the stimulation should pass precisely three times from c^* to d^* (i.e., through Ω). Or, which is more to the point, we

require that when a stimulus arrives at c (from d^* of Ω) the first two times, the consequences (in Φ) should be different from those that take place the third time. Hence Φ needs a memory that can distinguish the first two occasions from the third one, i.e., keep a count up to three.

With our present means this is best achieved by three periodic pulsers **PP**($\bar{\bar{1}}$), each of which is turned on (started) at the beginning of the count-period that it represents, and turned off (stopped) at its end. During that count-period, the corresponding **PP**($\bar{\bar{1}}$) effectively indicates the presence of the count-period by the (continuous) availability of its output stimulus. This stimulus must then be used to achieve the characteristic operations of the count-period in question. These are the following.

First count-period: A stimulus arriving at c is routed to d, and after this the **PP**($\bar{\bar{1}}$) of the first period is turned off and that of the second period is turned on.

Second count-period: A stimulus arriving at c is routed to d, and after this the **PP**($\bar{\bar{1}}$) of the second period is turned off and that of the third period is turned on.

Third count-period: A stimulus arriving at c is routed to b, and after this the **PP**($\bar{\bar{1}}$) of the third period is turned off.

The initial stimulus at a must, of course, be routed to c, and it must also turn on the **PP**($\bar{\bar{1}}$) of the first period.

The obvious way in which the continuous emissions of a **PP**($\bar{\bar{1}}$) can be used to induce the desired consequences that are characteristic of its count-period (cf. above), with the help of the stimulus at d that actuates the period in question, is this. Use a confluent state **C** as a coincidence organ by giving it the route from the output of the **PP**($\bar{\bar{1}}$) referred to above and the route from c as inputs (whose coincident stimulations it is to sense). Then use this **C** to initiate the characteristic responses enumerated above. [See cells *F14*, *F8*, and *F2* of Figure 23. Von Neumann thinks of the stimulus at d (i.e., c^*) as actuating a given count-period. This stimulus will pass through Ω and enter Φ at c (i.e., d^*).]

All these arrangements are shown in Figure 24a. Note that in this figure the inputs and the output of the **PP** are designated by $\bar{a}_+$, $\bar{a}_-$ and $\bar{b}$ (instead of a_+, a_- and b, as in Figs. 19g and 19h). The upper and the lower edges of the **PP** are protected by **U**'s, to prevent unwanted stimulations by **C**'s across these boundaries. It is convenient not to extend this protection to the left lower corner of each **PP** (to make room for a transmission channel there); that this is permissible is

not indicated by Figure 19h, but it is by Figure 19g. [See also Figures 16b and 18k.]

In Figures 24a and 24b the following simplifications are used: A straight line of states $\underline{0}$ is designated by a single arrow, and similarly for the three other directions. A connected area of states **U** is cross-hatched. The **PP** ($\bar{\bar{1}}$) are not shown in their true size (which is 4×15; cf. Fig. 19g); the area of the network of Figure 24a is therefore correspondingly contracted horizontally and vertically.

Figure 24a does not show how the transmission channels of the secondary input-output pair c, d and of the primary input a reach their proper endings at $c = d^*$, $d = c^*$ and a, respectively. (In the figure $c = d^*$, $d = c^*$ are shown; a is not shown but should be thought of as lying towards the lower left corner.) In the figure the transmission channels in question end at c_1' or c_2', at d', and at a_1' or a_2', respectively. The primary output b is shown reaching its proper ending. There is no difficulty in connecting d' with d, but there is a serious topological difficulty in connecting c_1' or c_2' with c and at the same time a_1' or a_2' with a (without disturbing the connections of b and of d in the process): It is clear that these paths must cross. Note that it is not a rigid requirement that a, b, c, d be at the positions indicated in the figure (regarding a, cf. the above remark); i.e., a, b, c, d could be moved around somewhat. However, it is necessary that a, b be on the left edge of the network and that c, d be on its right edge (cf. later [Fig. 39]), and this implies the topological difficulty referred to above, i.e., the necessity of crossing paths. We may therefore just as well leave a, b, c, d in the positions indicated above. We must solve the problem of how to cross the channel from c_1' or c_2' to c and that one from a_1' or a_2' to a without corruption of information by interference, i.e., without the information intended for one channel getting into the other one, and thereby causing malfunctions in the network.

The problem of crossing lines (which is a peculiarity of the 2-dimensional case; cf. Chapter 1 and also later) will be solved in a general form by the construction of a specific organ for this purpose later on [Sec. 3.6]. However, its present occurrence is particularly simple, and it does therefore not seem worthwhile to appeal to the general procedure referred to, but it is preferable to solve it ad hoc.

The main simplification present is that, while it is necessary to keep signals in the channel from c to c_1' or c_2' from getting into the channel from a to a_1' or a_2', the reverse is not necessary. Indeed, the first type of cross-talk between these channels would cause a pulse to cycle indefinitely between the responding organ Ω and the triple-

return counter Φ, thereby completely vitiating the operation. The second type of cross-talk, on the other hand, merely injects the original stimulus (from a) into the c_1'–c_2' channel, thereby furnishing a stimulus to each one of the three coincidence-sensing **C**'s (these are 2 squares to the right of the c_1'–c_2' channel at the left edge of the figure). An examination of the delays (cf. below) will show that at this time none of the **PP** ($\bar{\bar{1}}$) is turned on yet, and therefore none of the **C**'s in question receives the other stimulus required for a coincidence. Hence, the misdirected a-stimuli of this class are harmless.

Thus, we must only prevent c-stimuli from getting into the a_1'–a_2' channel. This can be achieved by the same coding-and-decoding trick that we will use subsequently for the general purpose referred to above [see Sec. 3.6]. That is, we will use a pulser to replace an a-stimulus by a (coded) sequence of stimuli, and protect a_1'–a_2' by placing a decoder in front of it which will respond only to this sequence of stimuli and not to a single stimulus. In this way an a-stimulus, in its expanded (coded) form, will be able to get through to a_1'–a_2'; while a c-stimulus, which remains single, will not be able to reach a_1'–a_2'. In addition, it is not worthwhile to appeal at this point to our previous constructions of pulsers and decoders, but it is simplest to make up the necessary primitive organs specifically for the present purpose.

The coding of an a-stimulus need only be its replacement by two stimuli, and it is somewhat more convenient to choose these as not immediately consecutive. That is, we are using $\overline{1}0\overline{1}$ rather than $\overline{11}$. This makes the coding, as well as the decoding operation quite simple. The arrangements that achieve this are shown in Figure 24b. (Note that Fig. 24b extends the lower end of Fig. 24a. Inspection of the two figures shows clearly what their common elements are, i.e., in what manner they must be superposed.) The position of a can be at a_1 or a_2. This stimulates the leftmost **C** which sends stimuli over two paths with a differential delay of 2 to its right-hand neighbor $\underset{\rightarrow}{0}$. Thus a sequence $\overline{1}0\overline{1}$ arrives here, if the stimulation came from a_1–a_2. Note that this $\underset{\rightarrow}{0}$ is also fed by a $\uparrow 0$ immediately under it, which represents the channel coming from c. Through this channel only a single $\overline{1}$ will come in. Thus, the coding of an a_1–a_2 stimulus (and the non-coding of a c-stimulus) has already taken place at this $\underset{\rightarrow}{0}$ (immediately to the right of the leftmost **C**). This stimulates the second **C**. From here this stimulation travels, as it should, unimpeded to c_2' (whether coded or not). On the other hand, two paths, with the differential delay 2, go from this (the second from the left) **C** to the next (the third from the left) **C**, which now acts as a coincidence

organ. That is, this last **C** will respond only if the previous **C** had accepted two stimuli with the compensating delay 2, i.e., a sequence $\overline{101}$. In other words, only a stimulus coming from a_1–a_2 (and not one coming from c) can effect (by these indirect means) a stimulation of this last **C**. This **C** coincides with that one in the lower left corner of Figure 24a, i.e., with the one adjacent to the a_1'–a_2' entries. Thus, the transmission from a_1–a_2 together with the exclusion from c, to a_1'–a_2', has been achieved, as desired.

Note that a_1' and a_2' have both been utilized (their **C** needed two accesses; cf. above), a may be at a_1 or a_2. We use a_1, since we want a from the left. Of c_1' and c_2' we have utilized c_2', and c_1' need not be considered further. Figures 24a and 24b having been fitted together, it is indicated to make a further addition to the left edge. Indeed, it is desirable to straighten it, and to protect its **C**'s (which could cause unwanted stimulations in adjacent transmission states) by a border of **U**'s. We add accordingly a border of **U**'s which fits the left edge of Figures 24a and 24b and contains the necessary channels from the new position of a to the old one, and from the new position of b to the old one. This is shown in Figure 24c, the position relatively to Figure 24b being indicated by the dashed line there. Finally, we fill the strip around the right upper corner of Figure 24a, marked with a dashed line, with **U**'s, to complete the rectangular shape.

This completes the construction. The areas **PP**$(\bar{\bar{1}})$ in Figure 24a have the width 15 and the height 4 according to Figure 19g. Hence the area of the network of Figure 24a spans a rectangle of width 21 and of height 26. This has the consequence that the (rectangular) area of the entire network (resulting from Figs. 24a–c) has the width 24 and the height 26. An abbreviated representation of this network is given in Figure 24d. Figures 24c and 24d are further contracted in comparison with Figures 24a and 24b.

The delay from a to a_1 is clearly 1 (cf. Fig. 24c); the delay from a_1 (through three **C**'s, and paying attention to the fact that because of the coding-decoding procedure used, the shorter path between the two first **C**'s must be combined with the longer one between the two last **C**'s, or vice versa, the same delay resulting in both ways) to $\bar{a}_+$ is 11 (cf. Fig. 24b for this and for several estimates that follow). This is also the delay to the output of the **10** immediately to the right of the last **C**, i.e., to the entrance of the straight channel leading to d. The delay through that channel to d (taking the true length of **PP**$(\bar{\bar{1}})$ into consideration, which is 15 according to Fig. 19g) is 18. The delay from $\bar{a}_+$ through **PP**$(\bar{\bar{1}})$ to $\bar{b}$ is 18 (cf. the end of Sec. 3.2.1). The delay from $\bar{b}$ to the first coincidence-**C** (2 squares to the right of

C at c_1'–c_2' in Fig. 24a; it must again be remembered that the true length of **PP**($\bar{\bar{1}}$) is 15 and that the true distance of b from the top of **PP**($\bar{\bar{1}}$) is 2; cf. Fig. 19g) is 22. Thus, the total delay from a to the first stimulation of Ω at $d = c^*$ is $1 + 11 + 18 = 30$. Furthermore, the total delay from a to the moment when the first coincidence-**C** becomes passable for stimuli coming from c_1'–c_2' (because it is being stimulated from the first **PP**($\bar{\bar{1}}$); cf. Fig. 24a) is $1 + 11 + 18 + 22 = 52$.

Note that it takes a pulse from a_1 that gets from the second **C** from the left into the c_1'–c_2' channel, a delay of 5 or of 7 (depending on which of the two available paths between the two first **C**'s is followed; cf. Fig. 24b) to get there (i.e., to emerge on the upper side of the second **C**). The delay from here to c_2' is 5 (since the true height of **PP**($\bar{\bar{1}}$) is 4; cf. Fig. 19g), and from there to the first coincidence-**C** (cf. Fig. 24a) is 3. Hence, the total delay from a to the first coincidence-**C** through this illegitimate channel, is $1 + 5 + 5 + 3 = 14$, or $1 + 7 + 5 + 3 = 16$. This delay increases by 9 for the second coincidence-**C**, and by 8 more for the third one. Hence it is at most $16 + 9 = 25$ and $25 + 8 = 33$ for these **C**, i.e., at most 33 overall. Now we saw that even the first coincidence-**C** becomes passable for stimulation from the c_1–c_2 channel only with a delay of 52 after the stimulation at a. (For the other coincidence-**C** this delay is, of course, still longer.) Hence this illegitimate stimulation comes too soon to cause trouble, confirming our earlier assertion to this effect.

[Von Neumann next considered the i'th response of Ω at $c = d^*$, for $i = 1, 2, 3$. He calculated the delays through Φ for each response and made sure that the internal timing was correct. The crucial point about timing is best explained in connection with Figure 23. Consider the pulse which constitutes the first response of Ω; let t be the time it arrives at cell *D14*. At time t gate *F14* is open and gates *F8* and *F2* are closed. This pulse will pass through gate *F14* and, among other things, enter input a_+^2 (at time $t + 9$) to start the second periodic pulser, which will in turn open gate *F8*. Will gate *F8* be opened in time to let the original pulse through to start the third periodic pulser, thereby creating a malfunction? A calculation of the delays along the two paths to cell *F8* shows that this will not happen: the original pulse reaches *F8* from the left at time $t + 12$, while the gating pulse reaches *F8* from the right at time $t + 49$. A similar calculation shows that there is no trouble at gate *F2*.]

This completes the discussion of the delays within Φ and the proof of the consistency of its functioning. We restate those of Φ's delay characteristics that are externally relevant: from the stimulation at

a to the first stimulation of Ω: 30; from the first response of Ω to the second stimulation of Ω: 66; from the second response of Ω to the third stimulation of Ω: 83; from the third response of Ω to the output at b (Fig. 24d): 59.

Let the delay from i'th stimulation of Ω to its i'th response be w_i ($i = 1, 2, 3$). (This is a property of Ω, not of Φ!) Then the total delay from a through Φ (and three times through Ω!) to b is $30 + w_1 + 66 + w_2 + 83 + w_3 + 59 = 238 + w_1 + w_2 + w_3$.

3.5 The $\overline{1}$ vs. $\overline{10101}$ Discriminator: Structure, Dimensions, and Timing

The next organ in this series is the $\overline{1}$ *vs.* $\overline{10101}$ *discriminator*. This organ has an input a and two outputs b, c. It performs a function that we have consciously refrained from postulating for the decoder (cf. the beginning of Sec. 3.3): it discriminates between two sequences, one of which is part of the other. To be specific, the arrival of a single stimulus at a causes an emission at b, provided that it is preceded by a no-stimulus sequence of sufficient length. The arrival of a sequence $\overline{10101}$ at a causes an emission at c (but not at b; i.e., it paralyzes the effect of the single stimulus that it contains; cf. above). . . .

The relation between the actuating sequences $\overline{1}$ (or rather $\overline{0 \cdots 01}$; cf. above) and $\overline{10101}$ at a and the response at b and c is freely timed; i.e., the delay between these is not prescribed at this point. . . .

The $\overline{1}$ vs. $\overline{10101}$ discriminator has the symbol Ψ.

[Von Neumann next developed his network for Ψ. For a reason to be explained, we have replaced his design of Ψ by a simpler one.

The discriminator Ψ is used in reading an arbitrary cell x_n of the infinite linear array **L** (Fig. 37). "Zero" is represented in cell x_n by the unexcitable state **U**, and "one" is represented by the quiescent but excitable state $\mathbf{T}_{030}$, which is an ordinary transmission state directed downward. Cell x_n is read by sending the sequence $\overline{10101}$ to input v_1 of connecting loop $\mathbf{C}_1$ and noting whether $\overline{1}$ or $\overline{10101}$ emerges from output w_1 of $\mathbf{C}_1$. The sequence $\overline{10101}$ travels down the upper half of $\mathbf{C}_1$ and enters cell x_n. If cell x_n is in state **U**, the sequence $\overline{1010}$ converts x_n into state $\mathbf{T}_{030}$, and the remaining $\overline{1}$ passes through x_n and travels down the lower half of $\mathbf{C}_1$ to output w_1. If x_n is already in state $\mathbf{T}_{030}$, the complete sequence $\overline{10101}$ travels around $\mathbf{C}_1$ and comes out at w_1. Hence a $\overline{1}$ at w_1 represents a "zero" at x_n, while a $\overline{10101}$ at w_1 represents a "one" at x_n. The $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ discriminates between these two cases.

At the time von Neumann designed his discriminator he did not know what other sequences might be fed into it. As the discriminator

is actually used in the read-write-erase unit **RWE** (Figs. 37 and 39), no other case arises. By taking advantage of this knowledge, and improving von Neumann's design in another respect, we can greatly simplify both his discussion and his design. For this reason we replace von Neumann's discriminator by Figure 25.

Assume as a rule of usage that Ψ is normally quiescent, but that on occasion either the sequence $\overline{10000}$ or the sequence $\overline{10101}$ enters input a, starting at time t, and that once one of these two sequences enters input a, no further stimuli enter input a until Ψ is quiescent again.

We will consider the two cases separately. In the first case, $\overline{1}$ enters input a at time t, travels along the path indicated by the long arrow, and is emitted from b at time $t + 40$. This $\overline{1}$ also enters the decoder $\mathbf{D}(\overline{10101})$, but it dies there.

In the second case the sequence $\overline{10101}$ enters a at times t through $t + 4$, producing two immediate effects. First, the sequence $\overline{10101}$ is decoded by $\mathbf{D}(\overline{10101})$, so that a single pulse emerges from output b^1 at time $t + 21$. Second, the sequence travels along the path from input a to output b, entering cell *J14* at times $t + 38$ through $t + 42$. This sequence would later be emitted from output b, except that it is blocked by the killing action from cell *J13* in the following way. The pulse from b^1 enters a^2 at time $t + 24$, so $\mathbf{P}(\overline{11111})$ sends stimuli into cell *J14* at times $t + 39$ through $t + 43$. This means that any pulse entering cell *J14* from cell *I14* during times $t + 38$ to $t + 43$ inclusive is lost. Since the sequence $\overline{10101}$ enters cell *J14* from the left at times $t + 38$ through $t + 42$, it is destroyed and there is no output from b. The pulse from b^1 at time $t + 21$ is emitted from output c at time $t + 25$.

Combining both cases we have, for the discriminator Ψ of Figure 25: if $\overline{10000}$ enters input a at times t through $t + 4$, a stimulus emerges from output b at time $t + 40$, and nothing comes from output c; while if $\overline{10101}$ enters input a at times t through $t + 4$, a stimulus emerges from output c at time $t + 25$, and nothing comes from output b. Hence Ψ performs the required discrimination.

We now summarize the characteristics of the discriminator Ψ of Figure 25 and von Neumann's design, putting the parameters for his design in parentheses. The $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ has width 10 (22), height 14 (20); the input a is 8 (1) cells above the bottom, the output b is 0 (6) cells above the bottom, and output c is 13 (18) cells above the bottom. The delay from input a to output b is 40 (86), and the delay from input a to output c is 25 (49). It is possible to make a much smaller $\overline{1}$ vs. $\overline{10101}$ discriminator than Figure 25, but

Figure 25 is in the spirit of von Neumann's design and is quite satisfactory for our purposes.

Discriminating between the two sequences $\overline{1}$ and $\overline{10101}$ is a special case of the general task of discriminating among binary sequences. Another instance of this general task occurs in the coded channel of the next section. Von Neumann solved the problem there by using sets of sequences such that no sequence bitwise implies (is covered by) any other, even if the sequences are shifted in time relative to one another. The sequences $\overline{1011}$, $\overline{1101}$, $\overline{1110}$ constitute such a set. For the sake of completeness we note that the general task of discriminating among binary sequences could be accomplished by a unit which "recognized" a given sequence and no other. An example of a recognizer is given in Figure 26.

The recognizer $\mathbf{R}(\overline{101001})$ of Figure 26 performs the following function. Suppose that at (relative) times 0 through 5 a sequence $i^1 i^2 i^3 i^4 i^5 i^6$ enters input a, and assume that this sequence is both preceded and followed by several zeros. Under these circumstances the recognizer $\mathbf{R}(\overline{101001})$ emits a stimulus from output b at time 48 if and only if the entering sequence is $\overline{101001}$ (i.e., i^1, i^3, i^6 are 1 and i^2, i^4, i^5 are 0). We will explain how the recognizer $\mathbf{R}(\overline{101001})$ accomplishes its purpose. The general principle (algorithm) for designing an arbitrary recognizer $\mathbf{R}(\overline{i^1 i^2 \cdots i^n})$ will be evident at the end of this explanation.

The following conditions obtain.

(I) The decoder $\mathbf{D}(\overline{101001})$ emits a pulse from output b^1 at time 23 if and only if i^1, i^3, i^6 are all 1.

(II) The pulser $\mathbf{P}(\overline{1101})$ emits a pulse from output b^2 at time 23 if and only if i^2 or i^4 or i^5 is 1.

There are three cases to consider.

(A) First case: the input sequence is $\overline{101001}$, i.e., i^1, i^3, i^6 are 1 and i^2, i^4, i^5 are 0. Nothing comes from b^2 or b^3, and the pulse from output b^1 is emitted from output b at time 48.

(B) Second case: i^1, i^3, i^6 are 1 and one or more of i^2, i^4, i^5 are 1. Pulses are emitted from both b^1 and b^2 at time 23. These pulses enter the confluent state *I5* at time 29, causing a pulse to enter input a^3 at time 31. The pulser $\mathbf{P}(\overline{11111})$ emits the sequence $\overline{11111}$, which enters cell *T1* at times 47 through 51 inclusive. The pulse from b^1 at time 23 zig-zags along rows *1* and *2*, entering cell *T1* at time 46, and is destroyed by the killing action of $\overline{11111}$ into cell *T1*. Hence nothing is emitted.

(C) Third case: not all of i^1, i^3, i^6 are 1. No pulse is emitted from

output b^1, and hence none from output b. If a pulse is emitted from b^2, it is blocked at confluent state *I5*.

This concludes our discussion of the recognizer $\mathbf{R}(\overline{101001})$.]

3.6 The Coded Channel

3.6.1 Structure, dimensions, and timing of the coded channel:

[In 3-dimensional space, wires can cross one another without intersecting, that is, without making any contact which transfers information from one to the other. In 2-dimensional space it is topologically necessary for communication channels to intersect, and so there is a problem of sending information down a channel in such a way that it does not appear in intersecting channels. This problem could be solved by adding a wire-crossing primitive. Such a wire-crossing primitive would itself involve extra states, and it would necessitate additional sensitized states for the direct (construction) process. Von Neumann solved the problem in his 29-state cellular system by means of a "coded channel."

Figure 27 shows an example of a coded channel constructed in accordance with von Neumann's algorithm. There are inputs a_1, a_2, a_3 and outputs b_1, b_2, b_3; each input a_i is associated with the corresponding output (or outputs) b_i. Thus a pulse into input a_2 will eventually appear at both b_2 outputs (not simultaneously) and nowhere else. The coded channel is made up of seven pulsers and seven decoding organs (all shown in reduced size), together with a "main channel" running from the output of $\mathbf{P}(\overline{10011})$ to the input of $\mathbf{D}(\overline{11001})$.

The coding is done with six sequences of length five such that none of these sequences bitwise implies (is covered by) any other. The sequences $\overline{11100}$, $\overline{11010}$, $\overline{11001}$, $\overline{10110}$, $\overline{10101}$, $\overline{10011}$ are associated with a_1, a_2, a_3, b_1, b_2, b_3, in that order. The way the sequences operate is best explained by means of an example. Suppose input a_2 is stimulated. This will cause pulser $\mathbf{P}(\overline{11010})$ to inject its characteristic $\overline{11010}$ into the main channel. This sequence will travel to the end of the main channel, but because it is distinct from every other sequence used it will affect only decoder $\mathbf{D}(\overline{11010})$. $\mathbf{D}(\overline{11010})$ will then send a pulse to pulser $\mathbf{P}(\overline{10101})$, which will inject its characteristics $\overline{10101}$ into the main channel. The sequence $\overline{10101}$ will travel to the end of the main channel, but because it differs from every other sequence used it will affect only the two decoders $\mathbf{D}(\overline{10101})$, both of which will emit pulses from their outputs b_2.

The inputs and outputs of the coded channel may be positioned in any order. It is because of this that two sequences are associated

with each input-output pair, the conversion from one sequence to its mate taking place at the top of Figure 27.

The inputs to the coded channel must be spaced sufficiently far apart in time to avoid corruption or cross talk. For suppose a_1 and a_2 were stimulated so their outputs $\overline{11100}$ and $\overline{11010}$ followed each other immediately in the main channel. The combined sequence $\overline{1110011010}$ contains the sequence $\overline{11001}$ which is assigned to input a_3, and hence which would operate $\mathbf{D}(\overline{11001})$ and eventually cause an output at b_3.]

Our third construction in this series is the *coded channel*. Up to now it has been our policy to give an exact and exhaustive description and discussion of each organ that we had in mind. In the present case, however, it is preferable to depart from this principle; i.e., it is much simpler and quite adequate for our purposes to proceed in a more heuristic fashion. We will therefore discuss more broadly what the need is that we wish to satisfy at this point, and by what means we can do this. We will then deal with the essential, prototype special cases, and develop a sufficient discussion, so that in our actual subsequent application (cf. later) the specific organ that is needed will emerge with little effort.

The coded channel performs a function which is necessitated by the peculiar narrowness of 2-dimensional space. Indeed, a logical network may easily require crossing of lines in a manner which cannot be accommodated in 2-dimensional space. For example, if five points a, b, c, d, e are given and every one of them is to be connected to every other, this cannot be done in 2 dimensions without intersections, as schematically shown in Figure 28a. The line connecting c to e is missing in this figure and cannot be inserted without an intersection. An actual instance of interconnections requiring intersections occurred during the construction of the triple-return counter (cf. the middle part of Section 3.4) in connection with the development of Figure 24b from Figure 24a.

It should be noted that this difficulty does not arise in 3-dimensional space. However, since all other phases of our constructions can be carried out in 2 dimensions, just as well as in 3, it is worthwhile to keep the dimensionality to the former, low value, if this is otherwise possible. For this reason we accept the subordinate difficulty of line-crossing in 2 dimensions and the necessity of special constructions to overcome it.

The obvious procedure would be to construct a special organ which performs the elementary act of line-crossing, as schematically shown in Figure 28b. A more detailed representation of this is given in Figure

28d: This organ has two inputs a, b and two outputs c, d. It is desired that a stimulation of input a elicit (with a suitable delay) a response at output d, and a stimulation at input b (again with a suitable, and possibly different delay) a response at output c. Note that the actual cyclical arrangement of a, b, c, d is essential: if they are arranged as shown in Figure 28e—or as shown schematically in Figure 28c—there would obviously be no difficulty whatever.

It is preferable to aim immediately at somewhat more than this. Indeed, if we only constructed organs as indicated in Figure 28d which effect a single line-crossing, we would have to combine large numbers of these in our subsequent constructions. This would lead to quite awkward geometrical configurations and make our general design operations rather inconvenient. It is therefore better to construct a multiple line-crossing organ, in the sense that we will describe more fully below.

A multiple line-crossing organ that would be universally useful may be described as follows. It is a rectangular area A, as shown in Figure 28f, on whose periphery there are various inputs and outputs. These inputs and outputs are shown in Figure 28f, and in Figures 28g–28k also, as bars. Every input is designated by a symbol a_ν, $\nu = 1, \cdots, n$; every output is designated by a symbol b_ν, $\nu = 1, \cdots, n$. There may be several a_ν with the same ν, and there may be several b_ν with the same ν. The order in which all of these a_ν and b_ν are arranged around the periphery of A may be prescribed in any manner whatsoever. It is desired that upon stimulation of an a_ν (i.e., of any one of the—possibly several—a_ν's with the ν under consideration) all the b_ν's (with the same ν) should respond, with appropriate (and not necessarily equal) delays.

However, for the purposes for which this device will be needed, a slightly weaker requirement will do equally well, and this requirement will be somewhat easier to implement, as will appear below. This weakening of the requirement is as follows. We define a certain cyclical sense of the periphery of A, as shown by the arrows in Figure 28g. We then break this cyclical arrangement; i.e., we change it into a linear one, by cutting it at the point designated by p—i.e., it is now thought to begin at p, circle the periphery of A in the direction of the arrows and end again at p. We now require that the stimulation of an a_ν should cause responses only in those b_ν (with the same ν) which lie ahead of the a_ν in question in the direction of the arrows. Finally we cut the periphery of A open at p; i.e., we replace it by an open line. This line may be straight or broken; i.e., it may consist of one or of more straight pieces, as shown in Figure 28h–28k. At

any rate, the inputs a_ν and the outputs b_ν remain attached to it as before, but we may also change (invert) the side of the line to which they are attached. (For the latter, cf. Fig. 28k.) Figures 28h–28k also show that we keep indicating the (previously cyclical, now linear) sense referred to above by arrows (as in Fig. 28g), and we replace the point p by p_1 (beginning) and p_2 (end).

The obvious way to achieve the functioning described above is by coding and decoding. To do this we correlate to every $\nu = 1, \cdots, n$ a suitable stimulus-no-stimulus sequence $\overline{i_\nu^1 \cdots i_\nu^m}$. (The length m of the sequence $\overline{i_\nu^1 \cdots i_\nu^m}$ could have been made dependent on ν, but this is not necessary.) Now we attach to every input a_ν a coding organ $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$, and to every output b_ν a decoding organ $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$. We connect all these inputs and outputs by a connected line of (ordinary) transmission states—$\underset{\rightarrow}{0}$ or $\uparrow 0$ or $\underset{\leftarrow}{0}$ or $\downarrow 0$, with the arrow each time pointing in the proper direction—following the path indicated by the arrows in the illustrative figure (any one of the Figs. 28h–28k that one wishes to use for picturing this procedure). To be more precise, the connecting links (for the entire system of a_ν's and b_ν's) must consist of such transmission states. This is also true for the places where the inputs (the a_ν's) are tied in. However, at the places where the outputs (the b_ν's) are tied in, the stimulus must be able to progress in two directions (namely along the subsequent transmission states and to the output that is attached at the point in question). Hence at each one of these places a confluent state $\mathbf{C}$ is required. This chain of (ordinary) transmission states and of confluent states will be called the *main channel* of the organ.

Figure 29 (which may be viewed as an elaboration of Figure 28i) shows this in more detail. This figure shows a particular, but typical, distribution of a_ν's and b_ν's ($\nu = 1, \cdots, n$) with $n = 2$. Note that in Figure 29 the input of $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ is at the distance of one square from the true input a_ν, and is designated by a_ν', while the output of $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ is designated by b_ν'. Also, the output of $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ is at the distance of one square from the true output b_ν, and is designated by b_ν'', while the input to $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ is designated by a_ν''. Note, furthermore, that each $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ or $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ may be rotated by any one of the four angles 0°, 90°, 180°, 270°, and, if necessary, also reflected about the horizontal or the vertical, against the standard arrangements in Figures 16g and 22c. Clearly, this calls merely for trivial transformations of the constructions of Figures 16 and 22, respectively.

We must now select the sequences $\overline{i_\nu^1 \cdots i_\nu^m}$, $\nu = 1, \cdots, n$. Each b_ν, i.e., each $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$, must respond to the $\overline{i_\mu^1 \cdots i_\mu^m}$ with

$\mu = \nu$, and to no other. In view of the properties of the $\mathbf{D}(\overline{i^1 \cdots i^m})$ {cf. the beginning of Sec. 3.3}, this means that no $\overline{i_\mu^1 \cdots i_\mu^m}$ with $\mu \neq \nu$ must contain $\overline{i_\nu^1 \cdots i_\nu^m}$. In order to avoid "misunderstandings" due to mis-positioning in time, this requirement must also be extended to the case when $\overline{i_\mu^1 \cdots i_\mu^m}$ and $\overline{i_\nu^1 \cdots i_\nu^m}$ are shifted relatively to each other (each one of these two sequences being both preceded and followed by a sufficiently long sequence of 0's).

This is certainly the case if the $\overline{i_\nu^1 \cdots i_\nu^m}$, $\nu = 1, \cdots, n$, are pairwise different, and if all of them begin with a stimulus and contain the same total number k of stimuli. We assume, therefore, that these conditions are fulfilled.

Hence our problem is to find n pairwise different sequences $\overline{i_\nu^2 \cdots i_\nu^m}$ (we need not consider i_ν^1, since it is 1), corresponding to $\nu = 1, \cdots, n$, each of which has the length $m - 1$ and contains precisely $k - 1$ ones. The number of (different) sequences of this kind is obviously

$$\binom{m-1}{k-1}.$$

Consequently the choice in the above sense is possible if and only if

$$(12') \qquad n \leqq \binom{m-1}{k-1}.$$

Therefore the only task remaining on this count is that of choosing m, k so that they fulfill condition (12′).

It would be unreasonable to choose k so that $k - 1$, too, fulfills condition (12′) {with the same m}. Hence

$$\binom{m-1}{k-1} > \binom{m-1}{k-2},$$

which means $k - 1 < m - k + 1$, $2k < m + 2$, and so $2k \leqq m + 1$, i.e.,

$$(13') \qquad k \leqq \frac{m+1}{2}.$$

Actually the choices $k = (m + 1)/2$ and $k = m/2$ are usually the practical ones.

These discussions make it clear how the construction of the coded channel must be effected in each specific special case that may arise. We will use for its abbreviated representation the schemata of Figures 28h–28k, with the arrows but without necessarily including the letters p_1, p_2, and with each cell entry or exit marked with its a_ν or b_ν, to the extent to which this is desirable.

There still remain some questions of detail that need to be considered.

All the organs $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ and $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ of Figure 29 have the same k (cf. Secs. 3.2.1 and 3.3 above), and therefore the same length (cf. Figs. 16g and 22c). As Figure 29 shows, the $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ and $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ are so arranged (oriented), that it is always this length that defines the width of the parallel strip along the main channel that they occupy. Hence this width is uniformly $2k$. Adding to it 1 for the main channel and 2 for two protecting strips of **U**'s along either side, we obtain the entire width of the parallel strip along the main channel, which forms the organ (i.e., the coded channel): $2k + 3$.

The heights of the various organs $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ and $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$, $u + 2$ with their various u, might vary a priori (cf. Figs. 16g and 22c). Inspection of the definition of the u of a $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ from its $u°$ (cf. (2′) and the remark before it in Sec. 3.2.1) shows that the u of the $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ can be made all equal (as the maximum of all $u°$). Let u' be the common u of all $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$. Inspection of the definition of the u of a $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ from its $u'°$ {cf. equation (11′) and the remark before it in Sec. 3.3} shows that the u of the $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ can be made all equal (as the maximum of all $u'°$); in addition, it is not hard to verify that all $u'°$ are automatically equal. At any rate, let u'' be the common u of all $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$.

As Figure 29 shows, the $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ and $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ are so arranged (oriented), that it is always these heights that define the distances of the organs $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ and $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ along the main channel. However, the distances obtained in this way are in any event only lower limits: whenever it is desired to increase the distance between two neighboring organs {from the $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$, $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ class}, it suffices to replace the single separating line of **U**'s (as shown in Fig. 29) by a suitable, larger number of such lines.

The next question relates to delays. The delay from the stimulation at a particular a_ν to the response at a particular b_ν is easily determined by inspection of Figure 29. The [total] delay from a_ν' through $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ to b_ν' is $d' = 2k + u' + 2$ (cf. the end of Sec. 3.2.1); the delay from a_ν'' through $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ to b'' is $d'' = 3k + u'' + m$ (cf. the end of Sec. 3.3). Let Δ be the distance along the main channel from the $\overset{0}{\rightarrow}$ (or $\uparrow 0$ or $\overset{0}{\leftarrow}$ or $\downarrow 0$) where the a_ν is tied in to the **C** where the b_ν is tied in (taking every **C** twice, and including the terminal **C**, but not the initial $\overset{0}{\rightarrow}$—or $\uparrow 0$ or $\overset{0}{\leftarrow}$ or $\downarrow 0$—into the count of squares, i.e., the distance). Then the total delay from the stimulation at a_ν to

the response at b_ν is $1 + d' + 2 + \Delta + 1 + d'' + 1 = \Delta + d' + d'' + 5$.

We must also consider the question of corruption by interference, which assumes here the following shape. Clearly the stimulation at an a_ν will cause a response at every b_ν (assumed to lie ahead in the direction of the arrows along the main channel). Also, the stimulation at an a_ν will not cause a response at any b_μ with $\mu \neq \nu$, provided that no other stimulations (at one or more a_λ's, with λ's among which $\lambda = \nu$ as well as $\lambda \neq \nu$ may occur) have taken place. The question is therefore: Can stimulations at several a_ν's (with repeated or different ν's) cause a response at any b_μ, where neither of the stimulations referred to above could have caused a response by itself? Or, to be more specific: What rules must we establish for the stimulations at the a_ν, to prevent such occurrences?

These rules will be given in the form of prescribing certain minimum delays between a stimulation at an a_ν and a simultaneous or subsequent stimulation at any other a_λ. (This includes $\lambda = \nu$ as well as $\lambda \neq \nu$; also for $\lambda = \nu$ and a subsequent stimulation the two a_ν may or may not be the same.)

Consider, accordingly, a b_μ. A response will take place there if its $\mathbf{D}(i_\mu^1 \cdots i_\mu^m)$ receives, at its a_μ'', a stimulus-no-stimulus sequence containing $\overline{i_\mu^1 \cdots i_\mu^m}$. Each a_ν can produce a sequence $\overline{i_\nu^1 \cdots i_\nu^m}$. What we must prevent is that a superposition of several shifted specimens of these should contain a $\overline{i_\mu^1 \cdots i_\mu^m}$ which did not actually occur (in the correctly shifted position) as one of them.

This could only occur if two shifted $\overline{i_\nu^1 \cdots i_\nu^m}$ (of different origin) contributed together (but neither of them separately!) $\geqq k$ stimuli to the same sequence $\overline{j^1 \cdots j^m}$. Let this sequence $\overline{j^1 \cdots j^m}$ have subsequences of lengths m', m'', respectively, in common with these two sequences $\overline{i_\nu^1 \cdots i_\nu^m}$. Then $m', m'' \geqq 1$, $m' + m'' \geqq k$. The distance of the beginnings of the two shifted sequences $\overline{i_\nu^1 \cdots i_\nu^m}$ must then be either $\leqq |m' - m''|$ (if they are both at the same end of the sequence $\overline{j^1 \cdots j^m}$) or $\leqq 2m - m' - m''$ (if they are at opposite ends of the sequence $\overline{j^1 \cdots j^m}$). The former is $\leqq k - 1$, the latter is $\leqq 2m - k$, and (since (12′) requires $k \leqq m$) $k - 1 \leqq 2m - k$. Hence the distance in question is at any rate $\leqq 2m - k$. Hence the occurrence in question is excluded if the distance (of the beginnings) of the two shifted sequences $\overline{i_\nu^1 \cdots i_\nu^m}$ is $> 2m - k$, i.e., $\geqq 2m + 1 - k$.

Consider therefore two inputs a_ν, a_λ, such that the output b_μ lies ahead of them in the direction of the arrows along the main channel.

(For the relationships between ν, λ, and between a_ν, a_λ, cf. above.) By interchanging a_ν, a_λ, if necessary, let a_λ lie ahead of a_ν in the direction of the arrows along the main channel. Let the distances from the tie-in points of a_ν, a_λ to that one of b_μ be Δ^I, Δ^{II}, respectively, and between the tie-in points of a_ν and of a_λ (taking every **C** twice, and including one, but not the other endpoint, into the count of squares, i.e., the distance) $\Delta^* = \Delta^I - \Delta^{II}$.

Now let a_ν, a_λ be stimulated at the times t^I, t^{II}, respectively. Then the sequences $\overline{i_\nu^1 \cdots i_\nu^m}$, $\overline{i_\lambda^1 \cdots i_\lambda^m}$, that are thus created, appear at a_μ'' at the times $t^I + 1 + d' + 2 + \Delta^I + 1$, $t^{II} + 1 + d' + 2 + \Delta^{II} + 1$, respectively. The difference of these is $(t^I + \Delta^I) - (t^{II} + \Delta^{II}) = (t^I - t^{II}) + \Delta^*$. Hence our above condition becomes $|(t^I - t^{II}) + \Delta^*| \geqq 2m + 1 - k$. This means, that either $(t^I - t^{II}) + \Delta^* \geqq 2m + 1 - k$, i.e.,

$$(14') \qquad t^I \geqq t^{II} + (2m + 1 - k - \Delta^*),$$

or $(t^I - t^{II}) + \Delta^* \leqq -(2m + 1 - k)$, i.e.,

$$(15') \qquad t^{II} \geqq t^I + (2m + 1 - k + \Delta^*).$$

At this point it is best to distinguish certain special cases, as follows.

First, $t^I \leqq t^{II}$, i.e., a stimulation (at a_ν, time t^I) is followed by a stimulation ahead of it in the direction of the arrows along the main channel (at a_λ, time t^{II}). First subcase: $\Delta^* < 2m + 1 - k$. In this case condition (14′) is unfulfillable and condition (15′) requires that the delay (from t^I to t^{II}) be $\geqq (2m + 1 - k) + \Delta^*$. Second subcase: $\Delta^* \geqq 2m + 1 - k$. In this case condition (14′) requires that the delay from (t^I to t^{II}) be $\leqq \Delta^* - (2m + 1 - k)$ and condition (15′) requires that it be $\geqq (2m + 1 - k) + \Delta^*$.

Second, $t^I \geqq t^{II}$, i.e., a stimulation (at a_λ, time t^{II}) is followed by a stimulation behind it in the direction of the arrows along the main channel (at a_ν, time t^I). Third subcase: $\Delta^* < 2m + 1 - k$. In this case condition (14′) requires that the delay (from t^{II} to t^I) be $\geqq (2m + 1 - k) - \Delta^*$, and condition (15′) is unfulfillable. Fourth subcase: $\Delta^* \geqq 2m + 1 - k$. In this case condition (14′) is automatically fulfilled and condition (15′) is unfulfillable.

These four subcases can now be summarized, forming a rule, whose observance excludes corruption by interference. This rule is given in what follows.

(16′) If there has been a stimulation at an input a_ρ, then a stimulation at an input a_σ is only permissible with a delay d $(\geqq 0)$, subject to the following conditions.
Let the distance between the tie-in points of a_ρ and of a_σ (taking every **C** twice, and including one, but not the other endpoint, into the count of squares, i.e., the distance) be Δ^* $(\geqq 0)$.
First case: a_σ lies ahead of a_ρ in the direction of the arrows along the main channel. Then either $d \geqq (2m + 1 - k) + \Delta^*$, or, if $\Delta^* \geqq 2m + 1 - k$, alternatively $d \leqq \Delta^* - (2m + 1 - k)$.
Second case: a_σ lies behind a_ρ in the direction of the arrows along the main channel. Then there is a limitation only if $\Delta^* < 2m + 1 - k$, in which case $d \geqq (2m + 1 - k) - \Delta^*$.

3.6.2 Cyclicity in the coded channel. To conclude the discussion of the coded channel, we come back to the question of cyclicity, which was briefly referred to early in Section 3.6.1.

The reason for interrupting the main channel at p in Figure 28g, that is, for not continuing it from p_2 to p_1 in Figures 28h–28k and 29, is clear. If such a connection existed, i.e., if the main channel were a closed loop, then a sequence $\overline{i_\nu^1 \cdots i_\nu^m}$ (injected by an a_ν into this channel) would keep circulating in it forever, i.e., keep stimulating each b_ν periodically. This is not wanted—we want a stimulus at an a_ν to stimulate each b_ν precisely once. However, there are various ways to circumvent this difficulty. We are giving in what follows one way that seems particularly simple.

Consider Figure 30a, which is the equivalent of Figure 28g, but without the interruption at p. We have nevertheless indicated a direction along the main channel by the arrows, but this is now to be viewed as cyclical. We have also put two points p_1, p_2 on the main channel, but these are not meant as interruptions or terminations. The unmarked bars on the main channel are the a_ν and b_ν, as described in connection with Figures 28f and 28g–28k. We now proceed to transform the structure indicated by Figure 30a in the following way.

In addition to the n index values $\nu = 1, \cdots, n$, we introduce n additional index values $\nu' = 1', \cdots, n'$. We may identify these with $n + 1, \cdots, 2n$, i.e., put $\nu' = \nu + n$, but this is not relevant. However, it is relevant that n has been replaced by $2n$ {e.g., in condition (12′)}.

We now cut the periphery open between p_1 and p_2, and attach the continuations q_1, p_1 and p_2, q_2 to the main channel at these two points, as shown in Figure 30b. Along the original main channel, i.e.,

p_1p_2, we leave the a_ν unchanged, but we replace each b_ν by the corresponding $b_{\nu'}$. This is indicated in Figure 30b by affixing an asterisk to each bar along the original main channel, i.e., on the portion p_1p_2.

Next we place outputs $b_1, \cdots, b_n$ (in this order) on the inner side of p_2, q_2 (the one turned towards p_1, q_1), and the inputs $a_{1'}, \cdots, a_{n'}$, (in this order) on the inner side of p_1, q_1 (the one looking toward p_2, q_2). We then connect each b_ν ($\nu = 1, \cdots, n$) directly to its corresponding $a_{\nu'}$. All this refers to the area in Figure 30b that is enclosed by dashed line and cross hatched. These arrangements are shown in detail on Figure 30c, which gives an enlargement of that area. Note that the endings p_1, q_1 and p_2, q_2 of the main channel are moved apart in Figure 30c by as much as needed to accommodate the outputs $b_1, \cdots, b_n$ and the inputs $a_{1'}, \cdots, a_{n'}$, that we introduced above for this region. More specifically, each $a_{\nu'}$ means that a $\mathbf{P}(\overline{i_\nu^1 \cdots i_\nu^m})$ is there, and each b_ν means that a $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^n})$ is there. It is for these that space (i.e., the necessary distance between p_1, q_1 and p_2, q_2) must be provided. The connection from b_ν to $a_{\nu'}$, shown in Figure 30c, therefore follows the pattern shown in Figure 29: the output b_ν'' of $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ is followed (by one square) by b_ν; the input $a_{\nu'}'$ of $\mathbf{P}(\overline{i_{\nu'}^1 \cdots i_{\nu'}^m})$ is preceded (by one square) by $a_{\nu'}$. The connection from b_ν to $a_{\nu'}$ is shown in Figure 30c (by an arrow) as a vertical channel (plausibly of {ordinary} transmission states $\downarrow 0$), but it suffices to make it a direct contact of b_ν and $a_{\nu'}$. In fact, the single **U** borders of $\mathbf{D}(\overline{i_\nu^1 \cdots i_\nu^m})$ between b_ν'' and b_ν and of $\mathbf{P}(\overline{i_{\nu'}^1 \cdots i_{\nu'}^m})$ between $a_{\nu'}$ and $a_{\nu'}'$ (cf. the analogs to this in Fig. 29) can be identified so that b_ν'' merges with $a_{\nu'}$, and b_ν with $a_{\nu'}'$.

The functioning of this organ, according to Figures 30b–30c is now easily analyzed. A stimulation at an a_ν (which must be one of the bars marked with an asterisk on the portion p_1p_2 of Fig. 30b) can only stimulate a b_ν. There exists precisely one b_ν, and this lies ahead of the a_ν in the direction of the arrows along the main channel, namely on the portion p_2q_2, as detailed in Figure 30c. Hence the b_ν will be stimulated, and from it the stimulus goes directly (along the arrow) to $a_{\nu'}$, on the portion q_1p_1, as detailed in Figure 30c. Thus the $a_{\nu'}$ is stimulated, which can only stimulate $b_{\nu'}$'s. All $b_{\nu'}$'s lie ahead of the $a_{\nu'}$, in the direction of the arrows along the main channel, namely, on the portion p_1p_2 of Figure 30b, i.e., among the bars marked with asterisks. To sum up, a stimulus at an a_ν (necessarily at a bar marked with an asterisk in Fig. 30b) will stimulate all the $b_{\nu'}$ (all among the bars marked with asterisks in Fig. 30b)—irrespective of their positions relative to each other on the portion p_1p_2 of Figure 30b. Also, after this has taken place the stimulus will die, i.e., no periodic repetition

will take place, since the main channel of Figure 30b is open (from q_1 to q_2, i.e., is not cyclical). Thus, from the point of view of Figure 30a, we have precisely the events that were intended.

[This concludes von Neumann's discussion of the coded channel. It will be recalled that he concluded his discussion of the non-cyclic coded channel (Sec. 3.6.1) with rule (16′), which concerns corruption resulting from two stimuli entering the coded channel too close together. It is natural to ask why he didn't state a corresponding rule for the cyclical coded channel of Figure 30 (Fig. 27 is an example). Von Neumann gave no reason, but it may be because he later constructed the control organ for his universal self-reproducing automaton with a non-cyclic coded channel; see Figure 37.]

DESIGN OF A TAPE AND ITS CONTROL

4.1 Introduction

[*4.1.1 Abstract.* In the present chapter von Neumann shows how to embed an indefinitely extendible tape and its control in his infinite cellular structure.

The following units are involved in the tape and its control:

(1) A linear array **L** for storing information: "zero" is represented in cell x_n by state **U** and "one" is represented by state $\downarrow 0$.
(2) A connecting loop $\mathbf{C}_1$ for reading an arbitrary cell x_n .
(3) A timing loop $\mathbf{C}_2$, used in modifying the length of the connecting loop $\mathbf{C}_1$.
(4) A memory control **MC**, used to control the operations of **L**, $\mathbf{C}_1$. and $\mathbf{C}_2$.
(5) The constructing unit **CU**, which controls **MC**.

Except for **CU**, all these units are shown in Figure 37, though not in correct proportion. See also Figure 50.

Von Neumann describes these units in a general way in the remainder of the present section. He develops the detailed operations for lengthening and shortening loops $\mathbf{C}_1$ and $\mathbf{C}_2$ and for writing in cell x_n in Section 4.2. He designs most of the memory control **MC** in Section 4.3; the design will be completed in Section 5.1.

Von Neumann developed his design in several stages, which he thought out as he proceeded. The final design operates as follows.

The constructing unit **CU** sends a pulse to the memory control **MC** signifying that cell x_n is to be read. This pulse causes the sequence $\overline{10101}$ to enter the connecting loop $\mathbf{C}_1$. The sequence $\overline{10101}$ then enters cell x_n with the following effect: if x_n is in state **U** the sequence $\overline{1010}$ changes it into state $\downarrow 0$ and $\overline{1}$ returns to **MC**, while if x_n is in state $\downarrow 0$ the whole sequence $\overline{10101}$ returns to **MC**. A $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ detects the output and informs the constructing unit **CU** whether x_n stored a "zero" or a "one." In either case cell x_n is left in state $\downarrow 0$.

The constructing unit **CU** then tells the memory control **MC**

whether the loop $\mathbf{C}_1$ is to be lengthened so as to pass through cell x_{n+1} or shortened so as to pass through cell x_{n-1}, and whether cell x_n is to be left in state **U** ("zero") or ↓0 ("one"). Loop $\mathbf{C}_1$ is used to time the lengthening (or shortening) of loop $\mathbf{C}_2$. Then loop $\mathbf{C}_2$ is used to time the lengthening (or shortening) of loop $\mathbf{C}_1$. The new bit of information is written in cell x_n while loop $\mathbf{C}_1$ is being lengthened (or shortened). At the end of the whole process the memory control **MC** sends a finish signal to the constructing unit **CU**.

We have indicated here only those functions of the constructing unit **CU** which concern the memory control **MC**. The primary purpose of **CU** is to carry out the construction of a secondary automaton whose description is stored in **L**. Thus the universal constructing automaton has two parts: the constructing unit **CU**, and an arbitrarily large memory and its control (**MC**, **L**, $\mathbf{C}_1$, and $\mathbf{C}_2$). See Section 1.6.1.2 and Chapter 5.]

4.1.2 The linear array **L**. We have reached the point where the subsidiary constructions are completed and our first major synthesis can be undertaken. This highly composite organ that we will now construct has a single purpose, but it will actually account for approximately half of the entire self-reproducing organism.

It is best, therefore, to stop here for a moment and make some general observations regarding the overall organization that is to be developed.

We will need an automaton that can carry out general logical functions. The specific way in which this is integrated into the effecting of self-reproduction will be worked out in detail later, but the qualitative need for a general logical automaton should be clear a priori. These matters were broadly discussed in Chapter 1. They were first brought up in the formulation of the questions (A)–(E) in Section 1.1.2.1; their elaborations ran through Sections 1.2 to 1.7, and they were particularly in the foreground in Sections 1.4 to 1.6. We will therefore, for the time being, take the need for a general logical automaton for granted and discuss the ways and means of constructing one.

We noted in Sections 1.2.1 and 1.4.2.3 that a general logical automaton is necessarily organized around two main parts. The first part is a network that can carry out the elementary logical functions ($+$, $\cdot$, and $-$; cf. the main discussion in Sec. 1.2.1), and by combining these can evaluate all propositional functions in logics (cf. the end of Sec. 1.2.1). The second part is an arbitrarily large (finite, but freely adjustable) external memory, and the network that is needed to control and to exploit that memory. We know that the first part,

the machine for propositional functions, can be constructed with relatively simple means; indeed, the principles that are involved in this construction were in continuous use in our past constructions. We will take this part up in detail later. We turn our attention first to the second part, i.e., to the arbitrarily large external memory and its ancillary network.

The arbitrarily large external memory's physical embodiment is the linear array **L** that was discussed in Sections 1.4.2.1–1.4.2.4. We saw there that it was desirable to form it from cells, each of which could be in any one of k preassigned states. It became clear in Section 1.4.2.1 that these states, in order to be convenient for their "notational" role, must be quasi-quiescent states, i.e., states like **U** or the unexcited forms of transmission or confluent states. For a variety of reasons it is most practical to use **U**'s and ordinary transmission states. As indicated in Section 1.4.2.4 we will use a binary notation; i.e., we put $k = 2$. Accordingly, we will use the state **U** and a suitable ordinary transmission state. The orientation of the latter has to bear a certain relation to the orientation of the linear array **L**. Specifically, in view of the way in which we will use **L**, it is desirable that the orientation of this transmission state be transversal to the direction of **L**. The latter will be horizontal. Accordingly, it will be found convenient to orient the transmission state vertically down, that is, to use the state ↓0. (↓0 represents in this case the unexcited form of that state, i.e., $\mathbf{T}_{030}$; cf. Sec. 2.8.2.) In order to have a firm relationship to the binary notation, we stipulate that **U** correspond to the digit 0, and ↓0 to the digit 1.

The linear array **L** should therefore be thought of as a sequence of cells x_n, $n = 0, 1, 2, \cdots$, which form a continuous horizontal line, and for each of which the two states **U**, ↓0 are available. With each x_n we associate a numerical variable ξ_n, which designates the binary digit represented by x_n. Thus, $\xi_n = 0$ indicates that x_n is **U**, $\xi_n = 1$ indicates that x_n is ↓0. Strictly, the length of **L** should be finite; i.e., the range of n should terminate, say immediately before N: $n = 0, 1, \cdots, N - 1$. In any case, it is best to assume that for sufficiently large n, $\xi_n = 0$. In view of this, the end of **L** merges into the field of **U**'s that we expect to find outside the specifically constructed organs. It is therefore not important in what way **L** is actually terminated.

In order to make this external memory **L** useful, the ancillary networks referred to above must be provided; that is, it is necessary to construct the means for "exploring" **L**. This "exploration" includes reading **L** at any prescribed place, and also altering it in any desired way at any prescribed place.

We are therefore passing to the discussion of these manipulations on **L**: reading and altering at assigned locations.

4.1.3 The constructing unit **CU** *and the memory control* **MC**.[1] Reading **L** at an assigned location, say n, means to observe the value of ξ_n. Altering it there means to change it from its present value, ξ_n, to its next value, ξ_n'.

It is convenient to use an index s ($s = 0, 1, 2, \cdots$) to enumerate the successive applications of this step. Hence the present value of a ξ_n will be designated by ξ_n^s, and its next value by ξ_n^{s+1}. Thus ξ_n^{s+1} replaces the ξ_n' that was used above.

The n involved in both operations (reading and altering), too, depends on s, and this dependence must be shown explicitly. Let therefore n^s be the n used in the above sense at step number s.

The number n^s may be specified absolutely or relatively. The latter will prove to be the preferable alternative. By this we mean the following. It is not convenient—and if logical generality is absolutely insisted on, it is not even possible—to limit the size of n (or of N; cf. above). Hence, the number of binary digits of n is not limited either. Consequently, n cannot be (or cannot be conveniently) held "inside" the logical automaton, i.e., in its first main part according to the partition defined in Section 4.1.2. On the other hand, it would be very inconvenient to hold n in the second main part, i.e., in the unlimited "outside" memory **L**. In other words, it is undesirable at this stage of our automaton development (although we will do it later on, when the integration of our automaton will have progressed further) to use **L** itself for "addressing" within **L**. These two undesirable alternatives exclude between them the possibility of direct addressing, i.e., of "absolute" specification of n. The plausible procedure is therefore to use "relative" specification of n. That is, each time when it becomes necessary to specify n, we will not do this directly, but rather by stating how the n to be used is related to the n used immediately before. It suffices to allow for changing n at each such unit time step by a single unit.[2] In other words, the n to be considered, n^s, will always be related to the one considered immediately before it, n^{s-1}, by

$$(17') \qquad n^s = n^{s-1} + \epsilon^s \qquad (\epsilon^s = \pm 1).$$

[1] [Von Neumann did not have titles for these two units but merely called them "**A**" and "**B**", though the units are not the same as his **A** and **B** of Sec. 1.6. His title for the present subsection was "The detailed functioning of **L**. The networks **A** and **B**. The function of **A** and its relationship to **B**."]

[2] Turing, "On Computable Numbers, With an Application to the Entscheidungsproblem."

The "relative" specification then consists of stating which of its two possible values ϵ^s assumes, i.e., whether $\epsilon^s = 1$ or $\epsilon^s = -1$.

Let us now reconsider the two main parts into which the general logical automaton was divided in Section 4.1.2. We can now make this division more precise while reformulating it with some changes. The first part is the network that carries out the elementary logical functions and combines them to form general logical propositional functions. Let this part be designated by **CU**. The second part is the external memory **L**, and the network that is needed to control and to exploit **L**. Let the latter be designated by **MC**. Thus the second part, in the sense of Section 4.1.2, is **L** plus **MC**. We can now define the respective functions of **CU** and **MC** more precisely than heretofore.

[Fig. 37 shows the relation of the memory control **MC** to the linear array **L**, the connecting loop $\mathbf{C}_1$, and the timing loop $\mathbf{C}_2$.]

We know that the function of the memory control **MC** is to read and to alter **L** at assigned locations. We saw above what this involves specifically. Given ϵ^s, **MC** must replace n^{s-1} by n^s according to equation (17′); it must sense the $\xi^s_{n^s}$; and given $\xi^{s+1}_{n^s}$, it must replace $\xi^s_{n^s}$ by $\xi^{s+1}_{n^s}$. The only portions of this definition that require further explanation are those relating to the "given ϵ^s" and the "given $\xi^{s+1}_{n^s}$". That is, we must specify by what processes ϵ^s and $\xi^{s+1}_{n^s}$ are to be obtained.

It is best to attribute this function (the obtaining of ϵ^s and $\xi^{s+1}_{n^s}$) to the constructing unit **CU**. The unit **CU** must form them as a function of its own previous state, and of the information obtained in the process. The latter is, of course, the reading of $\xi^s_{n^s}$ that is performed by the memory control.

The reading of $\xi^s_{n^s}$ by **MC** occurs after the formation of ϵ^s, i.e., of n^s {according to equation (17′)}, but before the formation of $\xi^{s+1}_{n^s}$. It must therefore affect the latter, but not the former. However, it is preferable to transpose these operations, i.e., to describe the formation of n^{s+1} from n^s, rather than that of n^s from n^{s-1}. This means replacing s by $s + 1$ in equation (17′), i.e., replacing (17′) by

$$n^{s+1} = n^s + \epsilon^{s+1} \qquad (\epsilon^{s+1} = \pm 1). \tag{18′}$$

Now we can define the functioning of units **CU** and **MC** as follows: **CU** first gives a "start" pulse to **MC**, thereby starting **MC** on step number s (if this is the step that has been reached at this point). The memory control **MC** then reads $\xi^s_{n^s}$ and communicates the result of this reading to the constructing unit **CU**. This new information changes the state of **CU**. The unit **CU** thereupon forms $\xi^{s+1}_{n^s}$ and ϵ^{s+1} and communicates these to **MC**. The unit **MC** now changes $\xi^s_{n^s}$ into

$\xi_{n^s}^{s+1}$; then **MC** forms n^{s+1} according to equation (18′) and establishes the same contact with n^{s+1} (i.e., with $x_{n^{s+1}}$ in **L**) that it had previously with n^s (i.e., with x_{n^s} in **L**). This concludes step number s. The memory control **MC** communicates this fact to **CU**. If it is so prescribed for the state in which **CU** is at that point, **CU** then gives the next "start" pulse to **MC**, thereby starting **MC** on step number $s + 1$, etc., etc.

For the time being, we wish to give only a schematic description of the constructing unit **CU**. It suffices therefore to say that **CU**, being a finite network, has only a finite number of states, say a. Let these states be enumerated by an index $\alpha = 1, \cdots, a$. Let the state of **CU** at the beginning of step number s be α^s.

Then the relevant facts involving **CU** and its relationship to **MC** are contained in these specifications:

(19′.a) If **CU** is in the state α^s, and **MC** communicates to it a value $\xi_{n^s}^s$, then **CU** goes over into the state α^{s+1}, given by

$$\alpha^{s+1} = A(\alpha^s, \xi_{n^s}^s).$$

(19′.b) If **CU** is in the state α^{s+1}, then it forms (and communicates to **MC**) the $\xi_{n^s}^{s+1}$ given by

$$\xi_{n^s}^{s+1} = X(\alpha^{s+1}).$$

(19′.c) If **CU** is in the state α^{s+1}, then it forms (and communicates to **MC**) the ϵ^{s+1} given by

$$\epsilon^{s+1} = E(\alpha^{s+1}).$$

(19′.d) **CU** delivers the "start" pulse for step number s to **MC** if and only if its state α^s lies in the subset S of the α.

Thus the three functions

(20′.a) $A(\alpha, \xi)$ $(\alpha = 1, \cdots, a;\ \xi = 0, 1;$ values of $A = 1, \cdots, a)$,

(20′.b) $X(\alpha)$ $(\alpha = 1, \cdots, a;$ values of $X = 0, 1)$,

(20′.c) $E(\alpha)$ $(\alpha = 1, \cdots, a;$ values of $E = \pm 1)$,

and the set

(20′.d) S (subset of the set of all $\alpha = 1, \cdots, a$),

contain the operational description of **CU** and of its relationship to **MC**, insofar as these are relevant for our immediate purposes. Later

on, we will describe **CU** in detail,[3] but for the present the above description suffices. Our present task is to perform complete and detailed construction of **MC**, including its connections with **CU**.

4.1.4 Restatement of the postulates concerning the constructing unit **CU** *and the memory control* **MC**. Let us restate the postulated functioning of **CU** and **MC**:

(1) **CU** and **MC** are assumed to have reached the beginning of step number s. **CU** is in the state α^s, **MC** is connected to the cell x_{n^s} in **L**.

(2) If α^s is not in S, then **CU** will not communicate further with **MC**. If α^s is in S, then **CU** gives a "start" pulse to **MC**. The "start" pulse initiates the events of step number s.

(3) **MC** reads the state of x_{n^s}, i.e., the value of $\xi^s_{n^s}$, and communicates this to **CU**.

(4) **CU** then forms α^{s+1} according to (19′.a), and goes into the state α^{s+1}.

(5) **CU** then forms $\xi^{s+1}_{n^s}$ and ϵ^{s+1} according to (19′.b) and (19′.c) and communicates these to **MC**.

(6) **MC** changes x_{n^s}, as required to change $\xi^s_{n^s}$ into $\xi^{s+1}_{n^s}$.

(7) **MC** then forms $n^{s+1} = n^s + \epsilon^{s+1}$ {cf. equation (18′)} and establishes its connection with $x_{n^{s+1}}$, relinquishing its connection with x_{n^s}. That is, it moves its connection within **L** one square forward if $\epsilon^{s+1} = 1$, and one square back if $\epsilon^{s+1} = -1$.

(8) Finally, **MC** gives a "completion" signal to **CU**.

(9) Now step number s is completed and step number $s + 1$ begins. The cycle therefore resumes with postulate (1) above, etc., etc.

Since we wish to give only a schematic description of **CU** at this stage (cf. Sec. 4.1.2), the purely formal statements of postulates (1), (2), (4), (5), concerning **CU** will suffice for the time being. The statements of postulates (1), (2), (3), (6), (7), (8), on the other hand, concern **MC**, and our present task is to carry out an effective and detailed construction of **MC**. Hence these last mentioned statements must be implemented by actual constructions. We will do this in Sections 4.1.5–4.1.7, which follow.

4.1.5 The modus operandi of the memory control **MC** *on the linear array* **L**. We must first define the nature of the "connection" between **MC** and a specific cell x_n in **L**, as used in postulate (1) {for $n = n^s$} and postulate (7) {for $n = n^{s+1}$} above. This connection must be such that it makes the "reading" of x_n by **MC** possible {cf. postulate (3), with $n = n^s$}, as well as the changing of x_n by **MC** {cf. postulate

[3] [Von Neumann never reached the part of the manuscript where he planned to design the constructing unit **CU**. See Chapter 5 below.]

(6), with $n = n^s$}. It must also be possible to move this connection, operating from **MC**, one square forward or backward {cf. postulate (7)}.

The linear array **L** is a linear sequence of cells x_n, $n = 0, 1, 2, \cdots$. We assume it to be horizontal, extending, and numbered, from left to right, as shown in Figure 31a. We assume furthermore that **MC** lies to the left of **L**, as shown in the same figure.

The obvious way for reading cell x_n {with $n = n^s$; cf. postulate (3) in Sec. 4.1.4} consists of having a line of ordinary transmission states leading to it from **MC**, and then back from it to **MC**. We place the line from **MC** to x_n on the upper side of **L** and the line from x_n back to **MC** on the lower side of **L**. Together with x_n these form a loop from **MC** to **MC**, to be called the *connecting loop* and to be designated $\mathbf{C}_1$, as shown in Figure 31b. Now cell x_n can be "read" by sending a stimulus into the $\mathbf{C}_1$ loop at its entry v_1, and observing what emerges at its exit w_1. If x_n is **U** (i.e., $\xi_n{}^s = 0$) then the stimulus will not be able to pass through x_n (i.e., no response will appear at w_1). If x_n is $\downarrow^0$ (i.e., $\xi_n{}^s = 1$) then the stimulus will pass through x_n (i.e., a response will appear at w_1).

Note that the stimulus coming from v_1 through $\mathbf{C}_1$ will nevertheless affect x_n if it is **U** (i.e., if $\xi_n{}^s = 0$). It will transfer x_n into the sensitized state $\mathbf{S}_\theta$ (cf. Sec. 2.8 for this and for all subsequent discussions of transitions between states). If nothing further happened, x_n would then go from the state $\mathbf{S}_\theta$ spontaneously through certain successive states, the complete sequence being

$$(21') \qquad \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_0 \rightarrow \mathbf{S}_{00} \rightarrow \mathbf{S}_{000} \rightarrow \mathbf{S}_{0000} = \mathbf{T}_{000} = \overset{0}{\rightarrow}.$$

Now, it is preferable to have x_n terminate in the same state in which it would have been otherwise (i.e., if $\xi_n{}^s = 1$ had held instead of $\xi_n{}^s = 0$), namely, the state $\downarrow^0$. The reason for this is that our subsequent operations with x_n can be organized more simply if they are known to start with a fixed state of x_n under all conditions. The state $\downarrow^0$ is $\mathbf{T}_{030}$, i.e., $\mathbf{S}_{010}$. Hence the original stimulus (which transferred x_n from **U** into **S**) should be followed by the stimulus-no-stimulus sequence $\overline{010}$. Hence a total sequence $\overline{1010}$ (at v_1) is called for.

Injecting $\overline{1010}$ at v_1 thus transforms x_n, in the case $\xi_n{}^s = 0$, successively according to

$$(22') \qquad \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_0 \rightarrow \mathbf{S}_{01} \rightarrow \mathbf{S}_{010} = \mathbf{T}_{030} = \downarrow^0.$$

Since the $\downarrow^0$ appears at the end, none of these stimuli is able to pass

through x_n, so no response appears at w_1. In the case $\xi_n^s = 1$, on the other hand, x_n is $\downarrow 0$; hence it is not modified, and all stimuli pass through it, so that the response at w_1 is $\overline{1010}$. To sum up, injecting $\overline{1010}$ at v_1 leaves x_n at the end in any case in the state, $\downarrow 0$ and it produces at w_1 no response or the response $\overline{1010}$, for $\xi_n^s = 0, 1$, respectively.

This procedure still has one shortcoming: the criterion of "no response at w_1," which appears here as the characteristic of $\xi_n^s = 0$, is meaningless if it is not known at what time this "no response" at w_1 is expected, since there is equally no response at w_1 at other times, when v_1 has not been "questioned" (stimulated). This means that the delay from v_1 to w_1 must be established by other means. This will be needed for other purposes, too, and it will be achieved with the help of the so-called timing loop (cf. later). However, for the present purpose the desired result can be secured more simply by a direct procedure, and we are therefore choosing this procedure.

Indeed, it suffices to add one more stimulus to the input sequence $\overline{1010}$ at v_1, i.e., to use the sequence $\overline{10101}$ there. The cell x_n will be in any event in the state $\downarrow 0$ before the last stimulus; hence this stimulus will pass, appear at w_1, and still leave x_n in the state $\downarrow 0$. Thus injecting $\overline{10101}$ at v_1 leaves x_n at the end in any case in the state $\downarrow 0$, and it produces at w_1 the response $\overline{1}$ or $\overline{10101}$ for $\xi_n^s = 0, 1$, respectively.

If the input $\overline{10101}$ at v_1 is followed by $\overline{00}$, then for $\xi_n^s = 0$ the $\overline{1}$ at w_1 is also followed by $\overline{00}$, and since this $\overline{1}$ is certainly preceded by $\overline{0000}$ (cf. above), we have for $\xi_n^s = 0$ the sequence $\overline{0000100}$ at w_1. This sequence cannot overlap with a sequence $\overline{10101}$ (which corresponds to $\xi_n^s = 1$). Hence the response at w_1 times itself and discriminates unambiguously between $\xi_n^s = 0$ and $\xi_n^s = 1$. The organ which effects this discrimination is our $\overline{1}$ vs. $\overline{10101}$ discriminator (organ Ψ of Fig. 25).

Thus the problem of reading x_n (with $n = n^s$) is solved, and the outline of the construction of a network to implement this procedure is clear. (For the actual construction, cf. later.)

[The operation of inserting the sequence $\overline{10101}$ is the first of a complicated sequence of operations involving the linear array **L**, the connecting loop $\mathbf{C}_1$, and the timing loop $\mathbf{C}_2$. The remaining operations are developed in the balance of the present section (4.1) and the next section (4.2), and are summarized in Figure 38 of Section 4.3.2. See also Figure 37.

When von Neumann's design is finished, the following sequence of operations will be used to read from and write in cell x_n of **L**, and to

lengthen (or shorten) the connecting loop $\mathbf{C}_1$ and the timing loop $\mathbf{C}_2$ preparatory to reading cell x_{n+1} (or cell x_{n-1}):

(1) Read cell x_n with the sequence $\overline{10101}$, leaving it in state $\downarrow 0$.

(2) Lengthen (or shorten) the timing loop $\mathbf{C}_2$, using the connecting loop $\mathbf{C}_1$ to time this operation.

(3) Lengthen (or shorten) the bottom of the connecting loop $\mathbf{C}_1$, using the timing loop $\mathbf{C}_2$ to time this operation.

(4) Change cell x_n to **U** if necessary.

(5) Lengthen (or shorten) the top of the connecting loop $\mathbf{C}_1$, using the timing loop $\mathbf{C}_2$ to time this operation.]

4.1.6 The connecting loop $\mathbf{C}_1$. The connecting loop $\mathbf{C}_1$, as defined above, must also handle the problem of altering x_n (with $n = n^s$; cf. postulate (6) in Sec. 4.1.4), that is, of transferring x_n from that state **U**, $\downarrow 0$, which corresponds to $\xi_n{}^s = 0, 1$, respectively, into that one which corresponds to $\xi_n{}^{s+1} = 0, 1$, respectively.

It is natural, as well as simplest, to perform this transformation of x_n immediately after reading it according to Section 4.1.5. However, we will see later that this conflicts with some other desiderata, and we will therefore depart from this arrangement. Nevertheless, it is instructive to carry out a preliminary discussion of the altering of x_n based on this assumption, to be corrected later on.

Consider, accordingly, the process of altering x_n, as if it took place immediately after the "reading" of x_n. At that moment x_n is certainly in the state $\downarrow 0$ (cf. above), which simplifies the task somewhat. Indeed, it is now only necessary to distinguish between two cases: $\xi_n{}^{s+1} = 1$, in which case x_n is already in the desired state and nothing need be done; $\xi_n{}^{s+1} = 0$, in which case x_n must be transferred from $\downarrow 0$ to **U**. (If x_n had not been altered from its original form, there would be three distinct cases, as follows. First, $\xi_n{}^s = \xi_n{}^{s+1}$ {$= 0, 1$; it is irrelevant which}, and nothing need be done. Second, $\xi_n{}^s = 1$, $\xi_n{}^{s+1} = 0$, and x_n must be transferred from $\downarrow 0$ to **U**. Third, $\xi_n{}^s = 0$, $\xi_n{}^{s+1} = 1$, and x_n must be transferred from **U** to $\downarrow 0$.)

Thus the only operational problem that has to be considered here, is that of transferring x_n from $\downarrow 0$ to **U**. Since $\downarrow 0$ is an ordinary transmission state, this cannot be done with ordinary stimuli. Hence we have the problem of bringing special stimuli into the proper contact with x_n.

This can be done in the following manner. Let a special transmission state be in direct contact with the first $\overset{0}{\rightarrow}$ in $\mathbf{C}_1$, i.e., the one next to v_1 and above x_0. This can be done by placing a state $\downarrow 1$ immediately above the $\overset{0}{\rightarrow}$ in question, i.e., at the asterisk in Figure 31b. Now a special stimulus from the asterisk will transfer the $\overset{0}{\rightarrow}$ in question

(i.e., the one above x_0) into **U**. It is desirable to transfer this further into the state $\overset{1}{\rightarrow}$, which is $\mathbf{T}_{100}$ (i.e., $\mathbf{S}_{011}$). Hence the original, special stimulus should be followed by the stimulus-no-stimulus sequence $\overline{1011}$. These could be ordinary or special stimuli, since both kinds have the same effect on the sensitized states. It is simplest to keep the entire sequence of one kind, i.e., of special stimuli only. This means, then, injecting a total sequence of special stimuli $\overline{11011}$ from the asterisk. Hence the square above x_0 will be successively transformed according to

$$(23') \qquad \overset{0}{\rightarrow} \rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_0 \rightarrow \mathbf{S}_{01} \rightarrow \mathbf{S}_{011} = \mathbf{T}_{100} = \overset{1}{\rightarrow}.$$

Now the square above x_0 is a $\overset{1}{\rightarrow}$. Hence a second sequence of special stimuli $\overline{11011}$ injected from the asterisk will pass through the $\overset{1}{\rightarrow}$ above x_0 as special stimuli and hit the next $\overset{0}{\rightarrow}$, i.e., the one above x_1. It will therefore put this square through the successive transformations of expression (23′), and in the end leave it as a $\overset{1}{\rightarrow}$. Now the squares above x_0 and x_1 are both in state $\overset{1}{\rightarrow}$. Hence a third sequence of special stimuli $\overline{11011}$ injected from the asterisk will pass through the two $\overset{1}{\rightarrow}$ above x_0 and x_1 as special stimuli and hit the next $\overset{0}{\rightarrow}$, i.e., the one above x_2. It will therefore put this square through the successive transformations of expression (23′), and in the end leave it in state $\overset{1}{\rightarrow}$. Now the squares above x_0, x_1, and x_2 are all in state $\overset{1}{\rightarrow}$. Clearly this process can be continued at will. After the injection of n sequences of special stimuli $\overline{11011}$ from the asterisk, the n squares above x_0, x_1, $\cdots$, x_{n-1} are all in state $\overset{1}{\rightarrow}$.

At this point the procedure should be changed, since the square above x_n must be transformed so that it will direct a special stimulus to x_n, and not to the next $\overset{0}{\rightarrow}$ (the one above x_{n+1}). That is, it should become a $\downarrow^1$, which is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$). Therefore a sequence of special stimuli $\overline{11110}$ should now be injected from the asterisk, transforming the square above x_n according to

$$(24') \qquad \overset{0}{\rightarrow} \rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_1 \rightarrow \mathbf{S}_{11} \rightarrow \mathbf{S}_{110} = \mathbf{T}_{130} = \downarrow^1.$$

Now we have a continuous chain of special transmission states from the asterisk to x_n; hence we can transform x_n. This is a transformation from $\downarrow^0$ to **U**, and a single special stimulus delivered from the asterisk will induce this transformation.

Thus we need the following system of sequences of special stimuli injected from the asterisk: n sequences $\overline{11011}$, followed by the sequence $\overline{111101}$ (this is, of course, $\overline{11110}$ and then $\overline{1}$). The subsequent condition of $\mathbf{C}_1$ and **L** is shown in Figure 31c.

It is not desirable to leave the part of $\mathbf{C}_1$ that has been so modified

(i.e., its upper line) in this condition. However, it is easy to return it from its altered condition (in Fig. 31c) to its original condition (in Fig. 31b), leaving x_n in its altered condition (which is **U**). Indeed, since the upper line of $\mathbf{C}_1$ now consists of special transmission states, ordinary stimuli will do this. These can be injected at v_1. The n first squares (above $x_0, x_1, \cdots, x_{n-1}$) should be transformed from $\overset{1}{\rightarrow}$ into $\overset{0}{\rightarrow}$, which is $\mathbf{T}_{000}$ (i.e., $\mathbf{S}_{0000}$). Hence n sequences of ordinary stimuli $\overline{110000}$ must be injected at v_1. The first one will transform the square above x_0 from $\overset{1}{\rightarrow}$ into $\overset{0}{\rightarrow}$, according to

$$(25') \quad \downarrow 1 \rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_0 \rightarrow \mathbf{S}_{00} \rightarrow \mathbf{S}_{000} \rightarrow \mathbf{S}_{0000} = \mathbf{T}_{000} = \downarrow 0.$$

The second one will do the same to the square above x_1, etc., etc., and the n-th will do the same to the square above x_{n-1}. The square above x_n can now be transformed from $\downarrow 1$ into $\downarrow 0$. This is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence a sequence $\overline{11010}$ of ordinary stimuli must be injected at v_1. This will transform the square above x_n according to

$$(26') \quad \downarrow 1 \rightarrow \mathbf{U} \rightarrow \mathbf{S}_\theta \rightarrow \mathbf{S}_0 \rightarrow \mathbf{S}_{01} \rightarrow \mathbf{S}_{010} = \mathbf{T}_{030} = \downarrow 0.$$

Thus we need the following system of sequences of ordinary stimuli injected at v_1 : n sequences $\overline{110000}$, followed by the sequence $\overline{11010}$. The subsequent condition of $\mathbf{C}_1$ and **L** is again that shown in Figure 31b, but with x_n being **U**.

The above discussions show that ordinary as well as special stimuli must be injected into $\mathbf{C}_1$. It is desirable to control both kinds by ordinary stimuli from memory control **MC**. This calls for no extra arrangements for the ordinary stimuli to $\mathbf{C}_1$; they can be injected directly at v_1. However, further arrangements are needed for the special stimuli to $\mathbf{C}_1$, which have to be injected from the asterisk of Figures 31b and 31c. Thus there is need for an organ that converts ordinary stimuli into special stimuli. We constructed such an organ earlier for the purposes of the periodic pulser. It is shown in Figures 18g and 18i. The arrangement of Figure 18i is more convenient. Its attachment to $\mathbf{C}_1$ and **L** is shown in Figure 31d. The input to this subsidiary organ is designated by u_1.

In this way we have a system where the original stimulations are uniformly made by ordinary pulses either at u_1 or at v_1. The former cause the injection of special pulses into $\mathbf{C}_1$; the latter inject directly ordinary pulses into $\mathbf{C}_1$—in both cases into the square above cell x_0.

We can now give the final form of the description of the treatment that transfers x_n from **U** into $\downarrow 0$, and leaves everything else in $\mathbf{C}_1$ and **L** in its previous condition:

(27'.a) Inject n sequences $\overline{11011}$ at u_1.

(27′.b) Inject a sequence $\overline{111101}$ at u_1.
(27′.c) Inject n sequences $\overline{110000}$ at v_1.
(27′.d) Inject a sequence $\overline{11010}$ at v_1.

4.1.7 The timing loop $\mathbf{C}_2$. The scheme of the previous subsection, as summarized in rules (27′.a–27′.d) is still incomplete in one respect. Rules (27′.a) and (27′.c) call for the n-fold repetition of certain operations, injecting $\overline{11011}$ and $\overline{110000}$ at u_1 and v_1, respectively. This can be done with the periodic pulsers $\mathbf{PP}(\overline{11011})$ and $\mathbf{PP}(\overline{110000})$, but these must then be turned off with the delays $5n$ and $6n$, respectively, after they have been turned on. How are these delays to be ascertained?

Prima facie there would seem to be the same difficulty, connected with the size of n, and its consequent unsuitability for storage "inside" **MC**, which we observed at the beginning of Section 4.1.3. This difficulty can be overcome by essentially the same device: relative instead of absolute definition of n. However, it is impractical to sense n, with our present arrangements, on **L** itself. It is therefore necessary to introduce a second "outside" organ to store n—or rather to express it in so explicit a manner that its sensing for the above purposes becomes immediately feasible. The best way to do this seems to be the introduction of another loop from **MC** to **MC**, whose length each way is n, or n plus a fixed amount. That is, we assume that there is, at a suitable distance below **L** and parallel to it, a sequence of n ordinary transmission states $\overset{0}{\rightarrow}$ leading out from **MC**, this sequence being terminated by an ordinary transmission state $\downarrow 0$, and immediately under this another line of $n + 1$ ordinary transmission states $\overset{0}{\leftarrow}$ leading back to **MC**. We position these below the squares $x_0, x_1, \cdots, x_n$ of **L**. This will be called the *timing loop* and designated $\mathbf{C}_2$, as shown in Figure 31e. Clearly a stimulus injected into the timing loop at its input v_2 will reappear at its output w_2 with a delay of $2n + 2$. It will appear later on that it is desirable to provide $\mathbf{C}_2$ with the same additional facilities for the injection of special stimuli, as were provided for $\mathbf{C}_1$. Hence we place again the equivalent of Figure 18i above the $\overset{0}{\rightarrow}$ under x_0, and designate its input by u_2. This arrangement defines the distance from $\mathbf{C}_1$ to $\mathbf{C}_2$ and the position of both relatively to **MC**, as shown in Figure 31f.

Since the delay from v_2 to w_2 is $2n + 2$, the delay three times around this loop is $6n + 6$. This differs from the delay $6n$, required in connection with operation (27′.c) of Section 4.1.6 by 6, which is a fixed amount (i.e., independent of n). Hence it can be used to define the delay called for in connection with operation (27′.c), subject only

to the insertion of suitable fixed delays, which is feasible "inside" **MC**.

The delay required in connection with operation (27′.a) of Section 4.1.6 is $5n$, and no integer multiple of $2n + 2$ differs from this by a fixed amount (i.e., by one that is independent of n). This difficulty can be circumvented by adding a $\overline{0}$ (i.e., a "no stimulus") to the sequence $\overline{11011}$ in operation (27′.a). This gives the sequence $\overline{110110}$; i.e., it replaces operation (27′.a) as follows:

(27′.a′) Inject n sequences $\overline{110110}$ at u_1.

Now operation (27′.a′), too, requires a delay $6n$; i.e., it can be treated precisely like operation (27′.c).

The sensing of three trips around the timing loop can be effected with the triple-return counter. This organ Φ is shown in Figure 23. Its output d must then be attached to v_2 and its input c to w_2. Let the delay from d to v_2 be δ_2' and the delay from w_2 to c be δ_2''. In the terminology of Section 3.4, then, Ω is the line from d to v_2, plus $\mathbf{C}_2$ from v_2 to w_2, plus the line from w_2 to c. Hence the delay from c through Ω to d is $\delta_2' + (2n + 2) + \delta_2''$. Consequently, according to the end of Section 3.4 and Figure 24d, the delay from a through Φ (and Ω) to b is $238 + 3(\delta_2' + (2n + 2) + \delta_2'') = 6n + (3(\delta_2' + \delta_2'') + 244)$. This exceeds the delay $6n$, called for in connection with rules (27′.a′) and (27′.c), by $3(\delta_2' + \delta_2'') + 244$, which is a fixed amount (i.e., independent of n). Hence this can be adjusted for by suitable fixed delays that can be provided "inside" the memory control **MC**.

4.2 Lengthening and Shortening Loops $\mathbf{C}_1$ and $\mathbf{C}_2$, and Writing in the Linear Array L

4.2.1 Moving the connection on **L**. There remains one more problem for our preliminary consideration: that of moving the connection of **MC** with **L** one square forward or backward (cf. postulate (7) in Sec. 4.1.4).

This means lengthening or shortening both lines of the connecting loop $\mathbf{C}_1$ by one square. Now we saw in Section 4.1.6 that the timing loop $\mathbf{C}_2$ is expected to have the same length as the connecting loop $\mathbf{C}_1$. Hence both lines of $\mathbf{C}_2$, too, must be lengthened or shortened by one square.

It is desirable to perform these operations at a time when $\mathbf{C}_1$ forms an uninterrupted loop, from v_1 to w_1. (The loop $\mathbf{C}_2$ is normally uninterrupted from v_2 to w_2.) This is the case when x_n is in state $\downarrow 0$. We saw at the end of Section 4.1.5 and the beginning of Section 4.1.6

that this can be guaranteed during the period between the reading and the altering of x_n . (The reading necessarily precedes the altering.) We will therefore perform the operations in question (lengthening or shortening each line of $\mathbf{C}_1$ and of $\mathbf{C}_2$ by one square) during this period.

Beyond this point the two alternatives ($\epsilon^{s+1} = 1$, lengthening, and $\epsilon^{s+1} = -1$, shortening) are better considered separately.

[The general process to be carried out is that of altering the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ at their ends, and perhaps modifying the contents of the storage cell x_n . The procedure for doing this was illustrated in Figure 14 of Section 2.8.3. The construction path is changed back and forth between a path of ordinary transmission states and a path of special transmission states. A path of ordinary transmission states is converted into a path of special transmission states by means of special stimuli, and a path of special transmission states is converted into a path of ordinary transmission states by means of ordinary stimuli.

The following inputs are used: u_1 and v_1 for $\mathbf{C}_1$, and u_2 and v_2 for $\mathbf{C}_2$. See Figures 31–37 and 39. A sequence of ordinary stimuli into u_1 changes the upper part of $\mathbf{C}_1$ into a path of special transmission states, and a sequence of ordinary stimuli into v_1 changes the upper part of $\mathbf{C}_1$ back into a path of ordinary transmission states. Similarly for u_2 , v_2 , and $\mathbf{C}_2$.

The conversion of a cell from an ordinary transmission state to a special transmission state (or vice-versa) takes six stimuli in the worst case (see Fig. 10). Hence to change the path from v_1 to x_n from ordinary to special states (or vice-versa) requires approximately $6n$ pulses. Now n is an arbitrary finite number, and hence cannot be stored in the finite automaton **MC** (or the finite automaton **CU**). Von Neumann ingeniously solved this problem by introducing the timing loop $\mathbf{C}_2$.

The timing loop $\mathbf{C}_2$ is attached to a triple-return counter Φ_2 ; see Figure 39. In the terminology of Section 3.4, the loop $\mathbf{C}_2$ is the responding organ Ω of the triple-return counter Φ_2 . Three times the delay around loop $\mathbf{C}_2$ is approximately the variable part of the needed delay $6n$. The desired sequence of pulses into u_1 or v_1 is obtained from periodic pulsers $\mathbf{PP}(\overline{i^1 i^2 i^3 i^4 i^5 i^6})$ turned on for approximately this period $6n$. Similarly, the triple-return counter Φ_1 of Figure 39 and the loop $\mathbf{C}_1$ are used to time the sequence of pulses needed to modify loop $\mathbf{C}_2$. Finite sequences of pulses are needed for changing the ends of $\mathbf{C}_1$ and $\mathbf{C}_2$ and writing in cell x_n ; these are easily obtained from pulsers.

Before the lengthening (or shortening) process begins, cell x_n has

been read by means of a sequence $\overline{10101}$ and has been left in the state $\downarrow 0$ (see Sec. 4.1.5). When von Neumann has finished his design the following sequence of operations will be used to lengthen (or shorten) the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ and write in cell x_n :

(1) Lengthen (or shorten) the timing loop $\mathbf{C}_2$ by means of periodic pulsers and pulsers feeding into inputs u_2 and v_2 . The length of time each periodic pulser is on is controlled by the triple-return counter Φ_1 to which loop $\mathbf{C}_1$ is attached as its responding organ Ω.

(2) Make the following modifications in loop $\mathbf{C}_1$ and cell x_n :
   (a) Lengthen (or shorten) the bottom of loop $\mathbf{C}_1$, gaining access to it through cell x_n so as not to disturb cells x_{n-1} or x_{n+1} .
   (b) Leave cell x_n in state $\mathbf{U}$ if a "zero" is to be stored; leave cell x_n in state $\downarrow 0$ if a "one" is to be stored.
   (c) Lengthen (or shorten) the top of loop $\mathbf{C}_1$.

   The stimuli required for this process are fed into inputs u_1 and v_1 from periodic pulsers and plain pulsers. The length of time each periodic pulser is left on is controlled by the triple-return counter Φ_2 to which the loop $\mathbf{C}_2$ is now attached as its responding organ Ω.

At the end of these two steps cell x_n is left in the desired state, the connecting loop $\mathbf{C}_1$ passes through cell x_{n+1} on the lengthening option and through cell x_{n-1} on the shortening option, and the timing loop $\mathbf{C}_2$ is of the same length as loop $\mathbf{C}_1$.

In Sections 4.2.2–4.2.4 von Neumann derived 31 formulas describing the sequences to be fed into loops $\mathbf{C}_1$ and $\mathbf{C}_2$ to accomplish the purpose just described. These formulas are all summarized in Table II on p. 235 below. They are numbered 28′.a through 31′.h in Sections 4.2.2–4.2.4 and renumbered 0.1 through 0.31 in Table II.

Table III on p. 236 below summarizes the periodic pulsers needed to produce sequences repeated approximately n times and the excess delays involved in timing these periodic pulsers by means of the triple-return counters Φ_1 and Φ_2 and the loops $\mathbf{C}_1$ and $\mathbf{C}_2$.

Table IV on p. 237 below summarizes the pulsers needed to produce the fixed sequences called for by the formulas.

The reader may find it helpful to refer to Tables II–IV while reading the rest of the present section.]

4.2.2 Lengthening on **L**. Consider first the case $\epsilon^{s+1} = 1$ (lengthening). In performing this operation, n (or $n + 1$) ordinary transmission states ($\overset{0}{\rightarrow}$ on the upper lines of $\mathbf{C}_1$ and $\mathbf{C}_2$, $\overset{0}{\leftarrow}$ on the lower lines of $\mathbf{C}_1$ and $\mathbf{C}_2$) must be transformed repeatedly into the corresponding special transmission states ($\overset{1}{\rightarrow}$, $\overset{1}{\leftarrow}$, respectively), and conversely. This requires arrangements like (27′.a′) and (27′.c) (for the $\rightarrow$ class),

and others similar to these (for the $\leftarrow$ class); hence there exists again the necessity of measuring delays $6n$ (or $6n$ plus a fixed amount) between the turning on and off of the periodic pulsers that are used. While loop $\mathbf{C}_1$ is being worked on, these delay measurements can be effected with the help of loop $\mathbf{C}_2$ (and the triple-return counter, together with suitable fixed delays; cf. Sec. 4.1.7). While loop $\mathbf{C}_2$ is being worked on, it is plausible to use loop $\mathbf{C}_1$ for the same purpose (with similar auxiliary equipment). This is a reason why $\mathbf{C}_1$ should at this time form an uninterrupted loop (from **MC** to **MC**), i.e., why cell x_n should be in state $\downarrow 0$ (cf. above). We know that this condition is satisfied when our present procedure begins; hence the operations on loop $\mathbf{C}_2$ should be the first ones that are undertaken (before those on loop $\mathbf{C}_1$).

Thus our first task is to lengthen both lines of loop $\mathbf{C}_2$ by one square. This is best begun by transforming the entire upper line of $\mathbf{C}_2$ (the squares of $\mathbf{C}_2$ under x_0, x_1, $\cdots$, x_n—their number is $n + 1$) into $\overset{1}{\rightarrow}$, i.e., from its original condition in Figure 32a into that of Figure 32b. Since these squares are originally $\overset{0}{\rightarrow}$, i.e., ordinary states, special stimuli are required. We saw in Section 4.1.6 {cf. the discussion of operation (23′)} that each transformation requires the stimulus-no-stimulus sequence $\overline{11011}$, or, according to Section 4.1.7 {cf. the passage from operation (27′.a) to operation (27′.a′)}, the stimulus-no-stimulus sequence $\overline{110110}$. Hence we have this requirement:

(28′.a) Inject $n + 1$ sequences $\overline{110110}$ at u_2.

Note that this calls for a **PP**$(\overline{\overline{110110}})$, with a delay $6n + 6$ between turning on and off. It is appropriate to attach a triple-return counter Φ (cf. Fig. 24) to loop $\mathbf{C}_1$, since the latter is now being used for timing. Hence we attach the output d of Φ to v_1 and the input c of Φ to w_1. Let the delay from d to v_1 be δ_1', and the delay from w_1 to c be δ_1''. The delay from v_1 through loop $\mathbf{C}_1$ to w_1 is $2n + 3$. In the terminology of Section 3.4, Ω is the path from d to v_1, plus $\mathbf{C}_1$ from v_1 to w_1, plus the path from w_1 to c. Hence the delay from d through Ω to c is $\delta_1' + (2n + 3) + \delta_1''$. Consequently, according to the end of Section 3.4 and Figure 24 the delay from a through Φ (and Ω) to b is $238 + 3(\delta_1' + (2n + 3) + \delta_1'') = 6n + (3(\delta_1' + \delta_1'') + 247)$. This exceeds the delay $6n + 6$, called for in connection with rule (28′.a), by $3(\delta_1' + \delta_1'') + 241$, which is a fixed amount (i.e., independent of n). Hence this can be adjusted by suitable fixed delays, that can be provided "inside" **MC**.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is **U**) into $\downarrow 1$, and the square under this (which is also **U**) into $\overset{0}{\leftarrow}$; i.e.,

we go from Figure 32b to Figure 32c. The state $\downarrow 1$ is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$) and the state $\overset{0}{\leftarrow}$ is $\mathbf{T}_{020}$ (i.e., $\mathbf{S}_{001}$). Hence the sequences $\overline{1110}$ and $\overline{1001}$ of special stimuli are needed, i.e.:

(28′.b) Inject the sequence $\overline{11101001}$ at u_2.

Now the lower line of the timing loop $\mathbf{C}_2$ is in its desired condition, and there remains the task of dealing with the upper line. This begins by transforming all of it, except its last square (i.e., the squares under $x_0, x_1, \cdots, x_n$—their number is $n + 1$) into state $\overset{0}{\rightarrow}$. This loop $\mathbf{C}_2$ is transformed from its condition in Figure 32c into that of Figure 32d. Since the cells to be changed are in state $\overset{1}{\rightarrow}$, ordinary stimuli are required. We saw in Section 4.1.6 {cf. the discussion of operation (25′)} that each transformation requires the stimulus-no-stimulus sequence $\overline{110000}$. Hence we have this requirement:

(28′.c) Inject $n + 1$ sequences $\overline{110000}$ at v_2.

This calls for a **PP**($\overline{110000}$), with a delay $6n + 6$ between turning on and off. The same triple-return counter Φ, attached to $\mathbf{C}_1$, that we used in connection with operation (28′.a), can take care of this delay too. The details are the same as in the discussion after operation (28′.a); i.e., the delay from a to b {cf. that discussion and Fig. 24} exceeds the desired delay by $3(\delta_1' + \delta_1'') + 241$, which, being fixed, calls for adjustments that can be provided "inside" **MC**.

Finally, we transform the square to the right of the last $\overset{0}{\rightarrow}$, (which is $\downarrow 1$) into $\downarrow 0$; i.e., we go from Figure 32d to Figure 32e. The state $\downarrow 0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequence $\overline{11010}$ of ordinary stimuli is needed:

(28′.d) Inject the sequence $\overline{11010}$ at v_2.

This completes the lengthening of the timing loop $\mathbf{C}_2$. We must now perform the corresponding operation on the connecting loop $\mathbf{C}_1$.

The state of loop $\mathbf{C}_1$ is shown in Figure 33a. It is best to begin the lengthening operation on the lower line of $\mathbf{C}_1$, the access to it being obtained through the upper line of $\mathbf{C}_1$ and the state $\downarrow 0$ at the place of x_n. We transform accordingly the entire upper line, except its last square (the squares above $x_0, x_1, \cdots, x_{n-1}$—their number is n) into $\overset{1}{\rightarrow}$. That is, we transform the upper line of $\mathbf{C}_1$ from its condition in Figure 33a into that of Figure 33b. Since these squares are $\overset{0}{\rightarrow}$ (i.e., ordinary states), special stimuli are required. Just as for operation (27′.a′) {or operation (28′.a)}, each transformation requires the sequence $\overline{110110}$. Hence:

(29′.a) Inject n sequences $\overline{110110}$ at u_1.

This calls for a **PP**($\overline{110110}$), with a delay $6n$ between turning on and turning off. The timing can be achieved with a triple-return counter tied to v_2, w_2 (with its d, c, respectively), precisely as in the discussion after operation (27′.a′). However, the length of $\mathbf{C}_2$ is now 1 unit greater than it was there; hence the delay through $\mathbf{C}_2$ is increased by 2, and the delay 3 times through $\mathbf{C}_2$ is increased by 6. Hence the fixed excess delay, which must be compensated by adjustments "inside" the memory control **MC**, is increased by 6; i.e., it is now $3(\delta_2' + \delta_2'') + 250$.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\downarrow 0$) into $\downarrow 1$, the square under this (which is also $\downarrow 0$) also into $\downarrow 1$, the square under this (which is $\overset{0}{\leftarrow}$) into $\overset{1}{\rightarrow}$ (these three are the square above, at, and under the place of x_n), and the square to the right of this (which is **U**) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 33b to Figure 33c. The state $\downarrow 1$ is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$), $\overset{1}{\rightarrow}$ is $\mathbf{T}_{100}$ (i.e., $\mathbf{S}_{011}$), $\overset{0}{\leftarrow}$ is $\mathbf{T}_{020}$ (i.e., $\mathbf{S}_{001}$). Hence the sequences $\overline{11110}$, $\overline{11110}$, $\overline{11011}$, $\overline{1001}$ of special stimuli are needed, i.e.:

(29′.b) { Inject the sequence $\overline{1111011110110111001}$ at u_1.

Now we may restore the squares at and below the place of x_n to their final condition. We must again get in through the upper line. Hence we transform all of it, except its last square (the squares above x_0, x_1, $\cdots$, x_{n-1}—their number is n) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 33c to Figure 33d. Since the affected part of the upper line now consists of special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(29′.c) Inject n sequences $\overline{110000}$ at v_1.

This calls for a **PP**($\overline{110000}$), with a delay $6n$ between turning on and turning off. It can be handled by the same timing arrangements used for operation (29′.a).

Next we transform the square to the right of the last $\overset{0}{\rightarrow}$ (which is $\downarrow 1$) into $\downarrow 0$, the square under this (which is also $\downarrow 1$) into $\downarrow 0$, and the square under this (which is $\overset{1}{\rightarrow}$) into $\overset{0}{\leftarrow}$ (these three are the squares above, at, and under the place of x_n); i.e., we go from Figure 33d to Figure 33e. The state $\downarrow 0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$), $\overset{0}{\leftarrow}$ is $\mathbf{T}_{020}$ (i.e., $\mathbf{S}_{001}$). Hence the sequences $\overline{11010}$, $\overline{11010}$, $\overline{11001}$ of ordinary stimuli are needed, i.e.:

(29′.d) Inject the sequence $\overline{110101101011001}$ at v_1.

Now we can carry out the lengthening of the upper line of $\mathbf{C}_1$. We begin by transforming the entire upper line (the squares above x_0, x_1, $\cdots$, x_n—their number is $n+1$) into $\overset{1}{\rightarrow}$; i.e., we go from Figure 33e to Figure 33f. Since the upper line now consists of ordinary transmission states, special stimuli will do this. Just as for operation (29′.a), each transformation requires the sequence $\overline{110110}$, i.e.:

(29′.e) Inject $n+1$ sequences $\overline{110110}$ at u_1.

This calls for a $\mathbf{PP}(\overline{\overline{110110}})$, with a delay $6n+6$ between turning on and turning off. It can be handled by the same timing arrangements used for operation (29′.a), except that the delay to be timed is now longer by 6. Hence the fixed excess delay, which must be compensated by adjustments "inside" the memory control **MC**, is decreased by 6; i.e., it is now $3(\delta_2' + \delta_2'') + 244$.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is **U**) into $\downarrow^0$; i.e., we go from Figure 33f to Figure 33g. The state $\downarrow^0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequence $\overline{1010}$ of special stimuli is needed, i.e.:

(29′.f) Inject the sequence $\overline{1010}$ at u_1.

Finally we transform the balance of the upper line (the squares above x_0, x_1, $\cdots$, x_n—their number is $n+1$) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 33g to Figure 33h. Since the affected part of the upper line now consists of special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(29′.g) Inject $n+1$ sequences $\overline{110000}$ at v_1.

This calls for a $\mathbf{PP}(\overline{\overline{110000}})$, with a delay $6n+6$ between turning on and turning off. It can be handled by the same timing arrangements used for operation (29′.e).

This completes the lengthening of the connecting loop $\mathbf{C}_1$, i.e., the entire first case, $\epsilon^{s+1} = 1$.

4.2.3 Shortening on **L**. Consider now the case $\epsilon^{s+1} = -1$ (shortening). The treatment runs, in its main outline, parallel to that of the first case. Hence the operations on $\mathbf{C}_2$ should again be undertaken first (before those on $\mathbf{C}_1$).

Thus our first task is to shorten both lines of $\mathbf{C}_2$ by one square. This is best begun by transforming the entire upper line of $\mathbf{C}_2$, except its last square (i.e., the squares under x_0, x_1, $\cdots$, x_{n-1}—their number is n) into $\overset{1}{\rightarrow}$, i.e., from its original condition in Figure 32a into that of Figure 34a. Since these squares are originally $\overset{0}{\rightarrow}$ (i.e., ordinary

states), special stimuli will do this. Just as for transformation (29′.a), each transformation requires the sequence $\overline{110110}$, i.e.:

(30′.a) Inject n sequences $\overline{110110}$ at u_2.

This calls for a $\mathbf{PP}(\overline{110110})$, with a delay $6n$ between the turning on and turning off. It can be handled by the same timing arrangements that take care of operation (28′.a), except that the delay to be timed is now shorter by 6. Hence the fixed excess delay, which must be compensated by adjustments "inside" the memory control **MC**, is increased by 6; i.e., it is now $3(\delta_1' + \delta_1'') + 247$.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\downarrow 0$) into $\downarrow 1$, and the square under this (which is $\overset{0}{\leftarrow}$) into **U**; i.e., we go from Figure 34a to Figure 34b. The state $\downarrow 1$ is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$). Hence the sequences $\overline{11110}$ and $\overline{1}$ of special stimuli are needed, i.e.:

(30′.b) Inject the sequence $\overline{111101}$ at u_2.

Now we restore the upper line of $\mathbf{C}_2$, except its last square (i.e., the squares under $x_0, x_1, \cdots, x_{n-1}$—their number is n) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 34b to Figure 34c. Since these squares are now special transmission states, ordinary stimuli will do this just as for operation (27′.c). Hence each transformation requires the sequence $\overline{110000}$, i.e.:

(30′.c) Inject n sequences $\overline{110000}$ at v_2.

This calls for a $\mathbf{PP}(\overline{110000})$, with a delay $6n$ between turning on and turning off. It can be handled with the same timing arrangements that take care of operation (30′.a).

Next we transform the square to the right of the last $\overset{0}{\rightarrow}$ (which is $\downarrow 1$) into **U**; i.e., we go from Figure 34c to Figure 34d. For this a single ordinary stimulus is needed, i.e.:

(30′.d) Inject a single stimulus at v_2.

At this point the shortening of $\mathbf{C}_2$ is effected, since the two rightmost squares have been transformed into **U**'s, but the rightmost square of what is left of the upper line (the one under x_{n-1}) is not in the desired state (it is $\overset{0}{\rightarrow}$ instead of $\downarrow 0$). We go on to correct this.

We transform the upper line, except its last square (the squares under $x_0, x_1, \cdots, x_{n-2}$—their number is $n - 1$) into $\overset{1}{\rightarrow}$; i.e., we go from Figure 34d to Figure 34e. Since these squares are now ordinary transmission states, special stimuli will do this. Just as for operation (29′.a), each transformation requires the sequence $\overline{110110}$, i.e.:

(30′.e) Inject $n - 1$ sequences $\overline{110110}$ at u_2.

This calls for a **PP**($\overline{110110}$), with a delay $6n - 6$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (30′.a), except that the delay to be timed is now shorter by 6. Hence the fixed excess delay, which must be compensated by adjustments "inside" **MC**, is increased by 6; i.e., it is now $3(\delta_1' + \delta_1'') + 253$.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\overset{0}{\rightarrow}$) into $\downarrow 0$; i.e., we go from Figure 34e to Figure 34f. The state $\downarrow 0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequence $\overline{11010}$ of special stimuli is needed, i.e.:

(30′.f) Inject the sequence $\overline{11010}$ at u_2.

Finally we transform the balance of the upper line (the squares under $x_0, x_1, \cdots, x_{n-2}$—their number is $n - 1$) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 34f to Figure 34g. Since the affected part of the upper line consists now of special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(30′.g) Inject $n - 1$ sequences $\overline{110000}$ at v_2.

This calls for a **PP**($\overline{110000}$), with a delay $6n - 6$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (30′.e).

This completes the shortening of $\mathbf{C}_2$. We must now perform the corresponding operation on $\mathbf{C}_1$.

The state of $\mathbf{C}_1$ is shown in Figure 33a. It is best to begin the shortening operation (just as we did the lengthening operation) on the lower line of $\mathbf{C}_1$, the access to it being obtained through the upper line of $\mathbf{C}_1$ and the $\downarrow 0$ at the place of x_n. We transform accordingly first the entire upper line, except its last square (the squares above $x_0, x_1, \cdots, x_{n-1}$—their number is n) into $\overset{1}{\rightarrow}$; i.e., from its condition in Figure 33a into that in Figure 35a. Since these squares are $\overset{0}{\rightarrow}$ (i.e., ordinary states), special stimuli are required. Just as for operation (29′.a), each transformation requires the sequence $\overline{110110}$, i.e.:

(31′.a) Inject n sequences $\overline{110110}$ at u_1.

This calls for a **PP**($\overline{110110}$), with a delay $6n$ between turning on and turning off. The timing can be achieved with the same means that were used after operation (29′.a). However, the length of $\mathbf{C}_2$ is now 2 less than it was there; hence the delay through $\mathbf{C}_2$ is decreased by 4 and the delay three times through $\mathbf{C}_2$ is decreased by 12. Therefore, the fixed excess delay, which must be compensated by adjustments

"inside" the memory control **MC**, is decreased by 12; i.e., it is $3(\delta_2' + \delta_2'') + 238$.

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\downarrow 0$) into $\downarrow 1$, the square under this (which is also $\downarrow 0$) into $\downarrow 1$, and the square under this (which is $\overset{0}{\leftarrow}$) into **U** (these three are the squares above, at, and under the place of x_n); i.e., we go from Figure 35a to Figure 35b. The state $\downarrow 1$ is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$). Hence the sequences $\overline{11110}$, $\overline{11110}$, $\overline{1}$ of special stimuli are needed, i.e.:

(31′.b) Inject the sequence $\overline{11110111101}$ at u_1.

Now we may restore the square at the place of x_n to its final condition. We must again get in through the upper line. Hence we transform first all of it, except its last square (the squares above x_0, x_1, $\cdots$, x_{n-1}—their number is n) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 35b to Figure 35c. Since the affected part of the upper line now consists of special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(31′.c) Inject n sequences $\overline{110000}$ at v_1.

This calls for a $\mathbf{PP}(\overline{110000})$, with a delay $6n$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (31′.a).

Next we transform the square to the right of the last $\overset{0}{\rightarrow}$ (which is $\downarrow 1$) into $\downarrow 0$, and the square under this (which is also $\downarrow 1$) also into $\downarrow 0$ (these two are the squares above and at the place of x_n); i.e., we go from Figure 35c to Figure 35d. The state $\downarrow 0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequences $\overline{11010}$, $\overline{11010}$ of ordinary stimuli are needed; i.e.:

(31′.d) Inject the sequence $\overline{1101011010}$ at v_1.

Now we can carry out the shortening of the upper line of $\mathbf{C}_1$. We begin by transforming the entire upper line, except its last square (the squares above x_0, x_1, $\cdots$, x_{n-1}—their number is n) into $\overset{1}{\rightarrow}$; i.e., we go from Figure 35d to Figure 35e. Since the affected part of the upper line consists now of ordinary states, special stimuli will do this. Just as for operation (29′.a), each transformation requires the sequence $\overline{110110}$, i.e.:

(31′.e) Inject n sequences $\overline{110110}$ at u_1.

This calls for a $\mathbf{PP}(\overline{110110})$, with a delay $6n$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (31′.a).

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\downarrow 0$) into **U**; i.e., we go from Figure 35e to Figure 35f. For this a single special stimulus is needed, i.e.:

(31′.f) Inject a single stimulus at u_1.

At this point the shortening of $\mathbf{C}_1$ is effected, since the two rightmost squares have been transformed into **U**'s, but the upper line is not in its desired state. We go on to correct this.

We transform the upper line, except its last square (the squares above $x_0, x_1, \cdots, x_{n-2}$—their number is $n-1$) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 35f to Figure 35g. Since these squares are now special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(31′.g) Inject $n-1$ sequences $\overline{110000}$ at v_1.

This calls for a **PP**$(\overline{\overline{110000}})$, with a delay $6n-6$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (31′.a), except that the delay to be timed is now shorter by 6. Hence the fixed excess delay, which must be compensated by adjustments "inside" the memory control **MC**, is increased by 6; i.e., it is now $3(\delta_2' + \delta_2'') + 244$.

Finally, we transform the square to the right of the last $\overset{0}{\rightarrow}$ (which is $\overset{1}{\rightarrow}$) into $\downarrow 0$; i.e., we go from Figure 35g to Figure 35h. The state $\downarrow 0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequence $\overline{11010}$ of ordinary stimuli is needed, i.e.:

(31′.h) Inject the sequence $\overline{11010}$ at v_1.

This completes the shortening of the connecting loop $\mathbf{C}_1$, i.e., the entire second case, $\epsilon^{s+1} = -1$.

4.2.4 Altering x_n in **L**. We must now reconsider the procedure of altering x_n, as discussed in Sections 4.1.6 and 4.1.7.

The assumption underlying the procedure of Sections 4.1.6 and 4.1.7 was that the altering of x_n follows immediately upon the "reading" of x_n. However, in Sections 4.2.1–4.2.3 we carried out the lengthening or the shortening of $\mathbf{C}_1$ and $\mathbf{C}_2$ as if it occurred between these two, i.e., as if this lengthening or shortening followed immediately upon the reading of x_n. These two assumptions are in conflict, and we now stipulate that the last mentioned assumption (i.e., the assumption of Secs. 4.2.1 and 4.2.3) is valid. Hence Sections 4.1.6–4.1.7 must be reconsidered.

The most natural procedure would be to review the changes that the lengthening or shortening has produced in the relevant structures

—i.e., to $\mathbf{C}_1$, on which Sections 4.1.6–4.1.7 operate, and to $\mathbf{C}_2$, which is used to time these operations—and to correct Sections 4.1.6–4.1.7 to account for these changes. However, it is simpler to analyze the procedures for the lengthening and the shortening themselves, and to see in each case where the operations of Sections 4.1.6–4.1.7, or equivalent ones, fit in best. It turns out that these insertions can be effected with the help of considerably simplified variants of Sections 4.1.6–4.1.7.

The conclusion reached at the beginning of Section 4.1.6 still holds: in the case $\xi_n^{s+1} = 1$ nothing need be done, while in the case $\xi_n^{s-1} = 0$, x_n must be transferred from $\downarrow 0$ to **U**. Hence we need to consider the latter case only. However, we must discuss the two main alternatives ($\epsilon^{s+1} = 1$, lengthening, and $\epsilon^{s+1} = -1$, shortening) separately.

Consider first the case $\epsilon^{s+1} = 1$ (lengthening). We must decide where to insert the process of altering x_n from $\downarrow 0$ into **U** into the evolution of $\mathbf{C}_1$ according to Figures 33a–33h. It is easily seen that it fits best after Figure 33e, involving a change of the evolution through Figures 33f–33h, i.e., of the steps (29′.e)–(29′.g).

We modify the passage from Figure 33e to Figure 33f by transforming one square less, i.e., into the passage from Figure 33e to Figure 36a. This reduces the number of iterations in operation (29′.e) by one:

(29′.e′) Inject n sequences $\overline{110110}$ at u_1.

This calls for a $\mathbf{PP}(\overline{110110})$ with a delay $6n$ between turning on and turning off. It can be handled by the same timing arrangements that take care of operation (29′.a).

Next we transform the square to the right of the last $\overset{1}{\rightarrow}$ (which is $\downarrow 0$) into $\downarrow 1$, and the square under this (which is also $\downarrow 0$) into **U** (these two are the squares above and at the place of x_n); i.e., we go from Figure 36a to Figure 36b. The state $\downarrow 1$ is $\mathbf{T}_{130}$ (i.e., $\mathbf{S}_{110}$). Hence the sequences $\overline{11110}$, $\overline{1}$ of special stimuli are needed, i.e.:

(29′.f′) Inject the sequence $\overline{111101}$ at u_1.

Now we transform the entire upper line (the squares above x_0, x_1, $\cdots$, x_n—their number is $n + 1$) into $\overset{0}{\rightarrow}$; i.e., we go from Figure 36b to Figure 36c. Since the upper line consists of special transmission states, ordinary stimuli will do this. Just as for operation (27′.c), each transformation requires the sequence $\overline{110000}$, i.e.:

(29′.g′) Inject $n + 1$ sequences $\overline{110000}$ at v_1.

This calls for a $\mathbf{PP}(\overline{110000})$ with a delay $6n + 6$ between turning

on and turning off. It can be handled by the same timing arrangements that take care of operation (29′.e).

Finally we transform the square to the right of the last $\overset{0}{\dashrightarrow}$ (which is **U**) into $\downarrow^0$; i.e., we go from Figure 36c to Figure 36d. The state $\downarrow^0$ is $\mathbf{T}_{030}$ (i.e., $\mathbf{S}_{010}$). Hence the sequence $\overline{1010}$ of ordinary stimuli is needed, i.e.:

(29′.h′) Inject the sequence $\overline{1010}$ at v_1.

This completes the discussion of the first case, $\epsilon^{s+1} = 1$.

Consider now the case $\epsilon^{s+1} = -1$ (shortening). We must decide where to insert the process of altering $\downarrow^0$ into **U** into the procedure of Section 4.3.2, i.e., into the evolution of $\mathbf{C}_1$ according to Figures 33a and 35a–35h. It is easily seen that it fits best after Figure 35c, but it affects only the step from there to Figure 35d, i.e., the step (31′.d).

Indeed, $\downarrow^0$ at the place of x_n was made from $\downarrow^1$; hence it suffices to make **U** instead. That is, the second sequence mentioned before step (31′.d) {it is $\overline{11010}$} must be replaced by a single stimulus. Hence operation (31′.d) is replaced by this:

(31′.d′) Inject the sequence $\overline{110101}$ at v_1.

Now the $\downarrow^0$ at the place of x_n is replaced by **U** in Figure 35d. After this the evolution through Figures 35e–35h, i.e., the steps (31′.e)–(31′.h′) can go on unchanged. In all Figures 35e–35h the only alteration will be the same replacement of the $\downarrow^0$ at the place of x_n by **U**, and this has no effect on the operations mentioned above.

This completes the discussion of the second case, $\epsilon^{s+1} = -1$.

4.3 The Memory Control MC

[*4.3.1 The organization and operation of* **MC**. Figure 37 is a schematic diagram of the memory control **MC** and the organs it controls: the linear array **L**, the connecting loop $\mathbf{C}_1$, and the timing loop $\mathbf{C}_2$. This figure is not drawn to scale; **MC** is actually 547 cells high and 87 cells wide. We will first describe the organization of **MC**, and then we will explain how **MC** operates.

The most important parts of **MC** are the read-write-erase unit **RWE** and its control **RWEC**. The unit **RWE** is shown in Figure 39 and developed in Section 4.3.3. The organs marked "0.6," "0.10," etc., in Figure 39 are the pulsers of Tables III and IV. The pulser marked "0.0" is a $\mathbf{P}(\overline{10101})$ which sends the sequence $\overline{10101}$ into input v_1 of loop $\mathbf{C}_1$ for the purpose of reading x_n; the result which emerges from output w_1 goes to the $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ. The triple-return counter Φ_1 has loop $\mathbf{C}_1$ as its secondary organ Ω,

and the triple-return counter Φ_2 has loop $\mathbf{C}_2$ as its secondary organ. The stimuli into **RWE** from the left come from decoders in $\mathbf{CC}_1$, and the stimuli going from **RWE** to the left go into pulsers in $\mathbf{CC}_1$. The unit $\mathbf{CC}_1$ of **RWE** is part of the coded channel of **MC**.

The coded channel of **MC** consists of $\mathbf{CC}_1$, $\mathbf{CC}_2$, $\mathbf{CC}_3$, **X**, **Z**, and the main channel. The main channel goes from the solid circle to the solid square. The unit $\mathbf{CC}_2$ contains decoders whose inputs come from the main channel and whose outputs go into **RWEC**. The unit $\mathbf{CC}_3$ contains pulsers whose inputs come from **RWEC** and whose outputs go to the main channel. The units **X**, **Z**, and $\mathbf{CC}_1$ contain both decoders receiving their inputs from the main channel and pulsers feeding their outputs into the main channel.

The read-write-erase control **RWEC** is shown in Figure 41 and developed in Section 4.3.5. It contains 16 control units **CO**; each **CO** contains a $\mathbf{PP}(\bar{\bar{1}})$ which is active while that **CO** is in control (see Fig. 40 and Sec. 4.3.4). The **RWEC** also contains four $\mathbf{PP}(\bar{\bar{1}})$ which are used to store the bit which is to be written in cell x_n.

The inputs and outputs of **RWEC** are labeled to correspond to the inputs and outputs of **RWE**. For example, a stimulus from $v_1 \cdot 1$ goes into a pulser of $\mathbf{CC}_3$ which sends a coded sequence into the main channel; this sequence is recognized by a decoder of $\mathbf{CC}_1$ which sends a pulse into input $v_1 \cdot 1$ of **RWE**, thereby stimulating the pulser labeled "0.0" to send the sequence $\overline{10101}$ into v_1 for reading cell x_n.

The area **Y** of **MC** is used only to transfer stimuli from outputs o_2, o_3, and o_5 of the constructing unit **CU** to pulsers in **X**, and to transfer a stimulus from a decoder in **X** to input i_3 of **CU**.

The area **W** has the following function. Each periodic pulser of **RWE** is to be on for about $6n$ units of time, where x_n is the cell of **L** which is being scanned. The triple-return counter Φ_1 (with $\mathbf{C}_1$ as its secondary organ) is to supply this delay when a periodic pulser is feeding loop $\mathbf{C}_2$, and the triple-return counter Φ_2 (with $\mathbf{C}_2$ as its secondary organ) is to supply this delay when a periodic pulser is feeding loop $\mathbf{C}_1$. But time is lost between each of these triple-return counters and its secondary organs. Moreover, the output of a triple-return counter ($\Phi_1 \cdot b$ or $\Phi_2 \cdot b$) cannot be used to stop a periodic pulser until it passes through $\mathbf{CC}_1$, the main channel, $\mathbf{CC}_2$, **RWEC**, $\mathbf{CC}_3$, the main channel again, and $\mathbf{CC}_1$ again. Altogether about 2000 units of time are lost in these two ways. Consequently, a periodic pulser of **RWE** must be turned on about 2000 units of time later than the triple-return counter associated with it. The exact amount of delay needed varies with each of the 16 control organs **CO** of **RWEC**. Von Neumann solved the problem by dividing the delay area **W** into

four parts and associating one part with each periodic pulser of **RWE**. Most of the delay needed is obtained by sending a pulse through this delay area. The variable part of the delay is obtained in the delay area **D** of each **CO** (Fig. 40).

We will next explain briefly how the memory control **MC** functions under the general direction of the constructing unit **CU** (Figs. 37 and 50). The basic operation of reading, writing, and lengthening (or shortening) the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ is carried out in two stages. First, **CU** sends a signal to **MC** telling it to read; **MC** then reads cell x_n and sends the result to **CU**. Second, **CU** sends signals to **MC** telling it what to write in cell x_n and whether to lengthen or shorten the loops $\mathbf{C}_1$ and $\mathbf{C}_2$; **MC** then executes these instructions and sends a completion signal to **CU**.

The first stage of the basic memory operation begins when a stimulus emerges from output o_1 of **CU** and enters the top of **RWEC**. It enters $\mathbf{CC}_3$ at $v_1 \cdot 1$, is coded by a pulser of $\mathbf{CC}_3$, enters the main channel, enters $\mathbf{CC}_1$, is decoded by a decoder of $\mathbf{CC}_1$, enters **RWE** at $v_1 \cdot 1$, and stimulates the pulser $\mathbf{P}(\overline{10101})$ of **RWE**; this pulser is labeled "0.0" in Figure 39a. The sequence $\overline{10101}$ enters $\mathbf{C}_1$ at v_1 and goes down the upper part of $\mathbf{C}_1$. If cell x_n is in state **U** (representing a "zero") the sequence $\overline{10101}$ changes x_n to state $\downarrow 0$ and a $\overline{1}$ emerges from exit w_1 of $\mathbf{C}_1$; if cell x_n is in state $\downarrow 0$ (representing a "one"), the complete sequence $\overline{10101}$ emerges from exit w_1 of $\mathbf{C}_1$. In both cases the output goes to the $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ of **RWE**. If x_n stored "zero," a stimulus emerges from output b of Ψ and goes via $\Psi \cdot b$, $\mathbf{CC}_1$, the main channel, $\mathbf{CC}_2$, $\Psi \cdot b$, and **RWEC** (near the top) to the input i_1 of **CU**. If x_n stored "one," a stimulus emerges from output c of Ψ and goes via $\Psi \cdot c$, $\mathbf{CC}_1$, the main channel, $\mathbf{CC}_2$, $\Psi \cdot c$, and **RWEC** (near the top) to the input i_2 of **CU**. The unit **CU** now knows the contents of cell x_n . This completes the first stage of the basic memory operation of **MC**.

To start the second stage of the basic memory operation of **MC** the constructing unit **CU** sends the following signals to the top of **MC**:

(1) A stimulus from output o_2 if "zero" is to be written in x_n ; a stimulus from output o_3 if "one" is to be written in x_n .

(2) A stimulus from output o_4 if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be lengthened; a stimulus from output o_5 if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be shortened.

We will trace the effects of these stimuli separately.

A stimulus from o_2 passes through area **Y** and stimulates a pulser in area **X**. This pulser emits a coded sequence which travels along the main channel and enters two decoders in $\mathbf{CC}_2$. These decoders

then turn on two **PP**$(\bar{\bar{1}})$ of **RWEC**. One of these **PP**$(\bar{\bar{1}})$ is shown at the top of Figure 41b; it is used to control the writing operation if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are lengthened. The other **PP**$(\bar{\bar{1}})$ is the upper **PP**$(\bar{\bar{1}})$ of Figure 41e; it is used to control the writing operation if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are shortened. Similarly, a stimulus from o_3 causes two **PP**$(\bar{\bar{1}})$ of **RWEC** to be activated. One of these **PP**$(\bar{\bar{1}})$ is shown in Figure 41c; it is used to control the writing operation if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are lengthened. The other **PP**$(\bar{\bar{1}})$ turned on by the signal from o_3 is the lower **PP**$(\bar{\bar{1}})$ of Figure 41e; it is used to control the writing operation if loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are shortened. Both **PP**$(\bar{\bar{1}})$ turned on by signals from o_2 (or o_3) are turned off at the end of the basic memory operation by the completion signal that goes to input i_3 of **CU**.

The signals from output o_4 of **CU** (signifying lengthening) and output o_5 of **CU** (signifying shortening) enter **MC** in different ways. A stimulus from o_4 enters the top of **RWEC** and turns on the first control organ, namely $\mathbf{CO}_1$. After control organ $\mathbf{CO}_1$ directs **RWE** to carry out certain operations, it is turned off and the control organ $\mathbf{CO}_2$ is turned on. Then $\mathbf{CO}_3$ and $\mathbf{CO}_4$ are used in a similar way. At this point there is a branch. If a "zero" is to be written in x_n, control organs $\mathbf{CO}_5$ and $\mathbf{CO}_6$ are used; while if a "one" is to be written in x_n, control organs $\mathbf{CO}_7$ and $\mathbf{CO}_8$ are used. When either $\mathbf{CO}_6$ or $\mathbf{CO}_8$ is finished, a pulse is emitted from the output i_3 below it. This pulse travels via $\mathbf{CC}_3$ and the main channel to do two things: first, it enters area **X**, passes through area **Y**, and enters input i_3 of **CU**, signifying that the basic operation of **MC** is finished; second, it enters $\mathbf{CC}_2$ at four places to turn off the two **PP**$(\bar{\bar{1}})$ of **RWEC** that were storing the bit to be written in x_n.

The signal from output o_5 of **CU** (signifying shortening) passes through **Y**, **X**, the main channel, and $\mathbf{CC}_2$, and enters the control organ $\mathbf{CO}_9$. Control organs $\mathbf{CO}_9$, $\mathbf{CO}_{10}$, $\mathbf{CO}_{11}$, $\mathbf{CO}_{12}$, $\mathbf{CO}_{13}$, and $\mathbf{CO}_{14}$ are used in that order. The output β from $\mathbf{CO}_{14}$ is gated by the **PP**$(\bar{\bar{1}})$ of Figure 41e. If a "zero" is to be written in x_n, the signal $v_1 \cdot 4$ is used; while if a "one" is to be written in x_n, the signal $v_1 \cdot 5$ is used. In either case the control organs $\mathbf{CO}_{15}$ and $\mathbf{CO}_{16}$ are then used. When $\mathbf{CO}_{16}$ is finished, a pulse is emitted from the output i_3 below it. As in the case of lengthening, this pulse does two things: it turns off the **PP**$(\bar{\bar{1}})$ of **RWEC** that were storing the bit to be written in x_n, and it enters input i_3 of **CU** to signify that the basic memory operation of **MC** is finished. At the end of a basic operation the memory control **MC** is left in its original state.

Table V summarizes the effect of the control organs of **RWEC** on the pulsers and periodic pulsers of **RWE**. We will illustrate how these

control organs work by tracing the action of $\mathbf{CO}_1$. The design of a **CO** is given in Figure 40; all **CO** are the same except for the length of the delay path $\mathfrak{B}$. The particular control organ $\mathbf{CO}_1$ is located at the top of **RWEC** (Fig. 41a). The state of the connecting loop $\mathbf{C}_1$ and the linear array **L** at the time $\mathbf{CO}_1$ takes control is shown in Figure 33a; loop $\mathbf{C}_1$ passes through cell x_n and consists entirely of ordinary transmission states. The state of the timing loop $\mathbf{C}_2$ at the time $\mathbf{CO}_1$ takes control is shown in Figure 32a; the effect of $\mathbf{CO}_1$ on $\mathbf{C}_2$ is shown in Figures 32b and 32c.

The action of $\mathbf{CO}_1$ is as follows. The stimulus from output o_4 of **CU** (signifying lengthening) enters input a of $\mathbf{CO}_1$ and accomplishes three tasks. First, it goes via the top and left side of $\mathbf{CO}_1$ to enter input a_+ of the alternate $\mathbf{PP}(\bar{\bar{1}})$ of $\mathbf{CO}_1$, thereby activating this periodic pulser, which will be on while $\mathbf{CO}_1$ is in control. Second, the input a of $\mathbf{CO}_1$ goes from output c of $\mathbf{CO}_1$ into input $\Phi_1 \cdot a$ of $\mathbf{CC}_3$; from there via $\mathbf{CC}_3$, the main channel, and $\mathbf{CC}_1$ to output $\Phi_1 \cdot a$ of $\mathbf{CC}_1$; and from there into input a of the triple-return counter Φ_1 of **RWE** (Fig. 39a), thereby starting this triple-return counter. Third, the input a of $\mathbf{CO}_1$ goes via the delay path $\mathfrak{B}$ of $\mathbf{CO}_1$ to output b of $\mathbf{CO}_1$; from there to input $u_2 \cdot a_+$ of $\mathbf{CC}_3$; thence via $\mathbf{CC}_3$, the main channel, and **Z** into the delay area **W**; through a portion of **W** and back through **Z** to the main channel; through $\mathbf{CC}_1$ to output $u_2 \cdot a_+$ of $\mathbf{CC}_1$; and from there to input a_+ of the periodic pulser $\mathbf{PP}(\overline{110110})$ in the lower part of **RWE** (Fig. 39b), thereby starting this periodic pulser. The sequence $\overline{110110}$ repeatedly enters input u_2 of loop $\mathbf{C}_2$. Each $\overline{110110}$ converts a cell of the upper part of $\mathbf{C}_2$ from $\underset{\rightarrow}{0}$ to $\underset{\rightarrow}{1}$ in the following way: $\bar{1}$ kills $\underset{\rightarrow}{0}$ to **U**, $\overline{1011}$ changes **U** to $\underset{\rightarrow}{1}$ according to the direct process of Figure 10, and the final $\bar{0}$ has no effect. Hence n occurrences of the sequence $\overline{110110}$ into u_2 will convert the upper part of loop $\mathbf{C}_2$ from the path shown in Figure 32a to the path shown in Figure 32b.

The production of n sequences $\overline{110110}$ is controlled by the triple-return counter Φ_1 of **RWE**, which uses the loop $\mathbf{C}_1$ as its secondary organ Ω. The output from b of Φ_1 is used to turn off the $\mathbf{PP}(\overline{\overline{110110}})$ of Figure 39b in the following way. The output from b of Φ_1 enters $\mathbf{CC}_1$ at $\Phi_1 \cdot b$, goes through $\mathbf{CC}_1$ to the main channel and to $\mathbf{CC}_3$, and enters input d of each of the control organs $\mathbf{CO}_1$, $\mathbf{CO}_2$, $\mathbf{CO}_9$, $\mathbf{CO}_{10}$, $\mathbf{CO}_{11}$, and $\mathbf{CO}_{12}$. All these **CO** are inactive except for $\mathbf{CO}_1$, so in every case except this one the effect of the input to d to a **CO** is only to attempt to turn off an alternate $\mathbf{PP}(\bar{\bar{1}})$ which is already off. This does no harm, as we saw at the end of Section 3.2.2. In the case of $\mathbf{CO}_1$, the input to d turns its $\mathbf{PP}(\bar{\bar{1}})$ off and also passes through the

confluent cell at the lower right of $\mathbf{CO}_1$ and exits at e and f. The "off" signal into a_- of $\mathbf{PP}(\overline{1})$ causes a stop at b too late to block the emission from e and f.

The output from e of $\mathbf{CO}_1$ enters $\mathbf{CC}_3$ at $u_2 \cdot a_-$; goes through $\mathbf{CC}_3$, the main channel, and $\mathbf{CC}_1$; leaves $\mathbf{CC}_1$ at $u_2 \cdot a_-$ and enters $\mathbf{PP}(\overline{110110})$ at a_-. In this way $\mathbf{PP}(\overline{110110})$ is shut off after it has emitted exactly n sequences $\overline{110110}$.

The output from f of $\mathbf{CO}_1$ goes to two different places. First, it enters $\mathbf{CC}_3$ at $u_2 \cdot 1$; it passes through $\mathbf{CC}_3$, the main channel, and $\mathbf{CC}_1$; and finally it leaves $\mathbf{CC}_1$ at $u_2 \cdot 1$ and enters the pulser $\mathbf{P}(\overline{11101001})$, which is marked "0.2" in Figure 39b. The sequence $\overline{11101001}$ then enters input u_2 of $\mathbf{C}_2$. The first half $(\overline{1110})$ of this sequence changes a **U** to ı, while the last half $(\overline{1001})$ of this sequence changes the next **U** to o. Hence loop $\mathbf{C}_2$ is left in the state shown in Figure 32c.

Second, the output from f of $\mathbf{CO}_1$ enters input a of $\mathbf{CO}_2$, thereby starting the operation controlled by $\mathbf{CO}_2$. The periodic pulser of **RWE** which is controlled by $\mathbf{CO}_2$ produces the result given in Figure 32d, and the pulser of **RWE** which is controlled by $\mathbf{CO}_2$ produces the result shown in Figure 32e. Thus the control organs $\mathbf{CO}_1$ and $\mathbf{CO}_2$ together bring about the lengthening of the timing loop $\mathbf{C}_2$. The control organ $\mathbf{CO}_2$ then passes control to $\mathbf{CO}_3$.]

4.3.2 Detailed discussion of the functioning of **MC**. We enumerated in Section 4.1.4 the specific functions of the memory control **MC**, namely: a purely descriptive static statement in (1), the start in (2), the substantive operations in (3) {reading x_n}, (6) {altering x_n}, and (7) {moving x_n , i.e., $n = n^*$}; and the completion in (8). {All these numbers refer to the listing in Sec. 4.1.4.} Postulate (1) calls for no action. The implementation of (2) {responding to a starting signal from **CU** to **MC**} and of (8) {delivering a completion signal from **MC** to **CU**}, is easy; we will attach these where they obviously belong, at the beginning of (3) and at the end of (8), respectively. Carrying out the other substantive operations (3), (6), and (7) is, of course, much more complicated; Sections 4.1.5–4.2.4 gave an outline of this. Now that this outline is completed, we must perform the necessary constructions in detail. In other words, we must fill in specific componentry to execute the operations enumerated and discussed in Sections 4.1.5–4.2.4. We proceed to do this in what follows.

We discussed operation (3) first in Section 4.1.5. We then took up operation (6) in Sections 4.1.6–4.1.7, but this was only a preliminary discussion; the final form was developed in Section 4.2.4. This form was actually meant to be meshed with the processing of operation (7). The latter was discussed in Sections 4.2.1–4.2.2. Thus the com-

plete discussion must begin with Section 4.1.5 and then continue with Sections 4.2.1–4.2.3, the latter being combined with Section 4.2.4.

Section 4.1.5 requires us to inject the sequence $\overline{10101}$ (with a guaranteed $\overline{00}$ thereafter) at v_1 and to feed the output of w_1 into the input a of the $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ of Figure 25. The outputs b and c of Ψ then indicate $\overline{1}$ (i.e., $x_n = \mathbf{U}$ and $\xi_n = 0$) and $\overline{10101}$ (i.e., $x_n = \downarrow 0$, and $\xi_n = 1$), respectively.

Sections 4.2.1–4.2.3, combined with Section 4.2.4, call for more complicated arrangements. Section 4.2.1 does not describe any specific operations. Section 4.2.2 postulates a definite sequence of operations: (28′.a)–(28′.d), and (29′.a)–(29′.g). Section 4.2.4 replaces operations (29′.e)–(29′.g) by operations (29′.e′)–(29′.h′). Section 4.2.3 postulates these operations: (30′.a)–(30′.g) and (31′.a)–(31′.h). Section 4.2.4 replaces operation (31′.d) by operation (31′.d′). Note that Sections 4.2.2 and 4.2.3 are alternative, depending on the value of ϵ^{s+1} (whether $\epsilon^{s+1} = 1$ or -1, respectively), i.e., on the first response of **CU** according to postulate (5) in Section 4.1.4. The insertion of Section 4.2.4 into Sections 4.2.2 and 4.2.3 is also conditional, depending on the value of $\xi_n{}^{s+1}$ (it takes place only if $\xi_n{}^{s+1} = 0$; cf. the beginning of Sec. 4.1.6), i.e., on the second response of **CU** according to postulate (5) in Section 4.1.4.

In order to describe the communications between **CU** and **MC**, and in view of the fact that we will not describe here in detail the internal functioning of **CU**, we must define certain specific inputs and outputs of **CU**.

The inputs of **CU** correspond to the signals that go from **MC** to **CU** according to postulates (1)–(9) in Section 4.1.4. These are the following:

(i_1) A signal according to (3), indicating that $\xi_n{}^s$ ($n = n^s$) has been read by **MC** and found to be 0; i.e., $x_n = \mathbf{U}$.

(i_2) A signal according to (3), indicating that $\xi_n{}^s$ ($n = n^s$) has been read by **MC** and found to be 1; i.e., $x_n = \downarrow 0$.

(i_3) The completion signal of **MC** according to (8).

For each one of these signals the symbol in parentheses which precedes it is the designation of the input of **CU** to which it is to go. That is, these inputs are i_1–i_3.

The outputs of **CU** correspond to the signals that go from **CU** to **MC** according to postulates (1)–(9) in Section 4.1.4. These are the following:

(o_1) The start signal to **MC** according to (2).

(o_2) A signal according to (5), indicating that $\xi_n{}^{s+1}$ ($n = n^s$) has been formed by **CU** and found to be 0.

(o_3) A signal according to (5), indicating that ξ_n^{s+1} ($n = n^s$) has been formed by **CU** and found to be 1.

(o_4) A signal according to (5), indicating that ϵ^{s+1} has been formed by **CU** and found to be 1.

(o_5) A signal according to (5), indicating that ϵ^{s+1} has been formed by **CU** and found to be -1.

For each one of these signals the symbol in parentheses which precedes it is a designation of the output of **CU** from which it is to come. That is, these outputs are o_1–o_5.

Our discussion above further shows that the signals from o_4 and o_5 are immediately effective on **MC**: they determine whether **MC** enters the action cycle (28′.a)–(28′.d), (29′.a)–(29′.g) {with a possible replacement of (29′.e)–(29′.g) by (29′.e′)–(29′.h′)} or the action cycle (30′.a)–(30′.g), (31′.a)–(31′.h) {with a possible replacement of (31′.d) by (31′.d′)}. Obviously, the signal from o_1, too, is immediately effective on **MC**: it is a start signal.

On the other hand, the signals from o_2 and o_3 are not immediately effective on **MC**: they determine whether the replacements referred to above will be made in the action cycles generated by o_4 and o_5. (o_3 causes the replacement to be omitted; o_2 causes it to occur.) Consequently, o_2 and o_3 cannot act directly on the modus operandi of **MC**. They must, instead, activate two (alternative) memory organs in **MC** which can then effect the operations of **MC** when they reach those points where this is required. Let the memory organs activated by o_2 and o_3 be designated by α_0 and α_1, respectively.

These considerations are summarized in Figure 38, which shows schematically the logical structure of the procedure that **MC** is to follow.

In this figure the arrows (horizontal and vertical) indicate the path of stimulation. The double horizontal lines separate areas within which the stimulation passes entirely through the channels in **MC** to be constructed, while these double lines themselves are crossed by processes that take place outside these channels. (As a rule these processes take place in **CU**, but in one case, that of the second double line from above, they take place within the discriminator Ψ.)

The single horizontal lines separate alternatives; i.e., the two processes at the two sides of such a single horizontal line, but between a pair of double horizontal lines, represent alternatives that are in fact mutually exclusive.

Each bracketed equation on the left margin indicates a value of ξ_n^s or ξ_n^{s+1} or ϵ^{s+1} that is characteristically associated with the alternative shown next to the equation in question.

The figure contains seven groups of operations, referred to by the numbers {(28′.a)–(31′.h)} by which they were designated in Sections 4.2.2–4.2.4. These are restated explicitly in Table II.

We have given these operations new designations (0.1)–(0.31), which are shown in Table II, for the sake of convenience, in juxtaposition with the old designations (28′.a)–(31′.h). Among the 31 operations (0.1)–(0.31) there are 16 with an asterisk and 15 without. The former are repetitions (n or $n+1$ or $n-1$ repetitions; cf. there), hence they have to be executed by suitable periodic pulsers (**PP**'s), for whose turning on and turning off arrangements with appropriate delays must be made. The latter are single operations; hence they call for ordinary pulsers (**P**'s) only. Two of these {(0.19) and (0.29)} require even no **P**, since they call for the injection of single stimuli only.

The turning on and turning off and concomitant delay arrangements for the **PP** (for the 16 operations with an asterisk) were discussed in Sections 4.2.2–4.2.4. In each case this required using a triple-return counter Φ (cf. Fig. 23). The use of this organ for this purpose was introduced in Section 4.1.7, after operation (27′.a′), and in Section 4.1.9, after operation (28′.a). In the first case Φ had to be attached to $\mathbf{C}_2$, i.e., its output d to v_2 and its input c to w_2. In the second case Φ had to be attached to $\mathbf{C}_1$, i.e., its output d to v_1 and its input c to w_1. The subsequent discussions in Sections 4.2.2–4.2.6 showed that these were the only two forms in which Φ was required. We therefore provide two triple-return counters, Φ_1 and Φ_2, the first one being attached to $\mathbf{C}_1$ (i.e., its input $c = c_1$ to w_1 and its output $d = d_1$ to v_1), and the second one to $\mathbf{C}_2$ (i.e., its input $c = c_2$ to w_2 and its output $d = d_2$ to v_2). The notations introduced in Section 4.2.2 after operation (28′.a), and in Section 4.1.7 after operation (29′.a′), apply to Φ_1 and Φ_2, respectively. Thus δ_1'' is the delay from w_1 to c_1 and δ_1' is the delay from d_1 to v_1 (these refer to Φ_1), while δ_2'' is the delay from w_2 to c_2 and δ_2' is the delay from d_2 to v_2 (these refer to Φ_2).

Of the 16 operations with asterisks, 6 have to be timed by Φ_1 {(0.1), (0.3), (0.16), (0.18), (0.20), (0.22)} and 10 have to be timed by Φ_2 {(0.5), (0.7), (0.9), (0.11), (0.13), (0.15), (0.23), (0.25), (0.28), (0.30)}. Of the 6 in the first class, 3 require a **PP**($\overline{\overline{110110}}$) with its output attached to u_2 {(0.1), (0.16), (0.20)} and 3 require a **PP**($\overline{\overline{110000}}$) with its output attached to v_2 {(0.3), (0.18), (0.22)}. Of the 10 in the second class, 5 require a **PP**($\overline{\overline{110110}}$) with its output attached to u_1 {(0.5), (0.9), (0.13), (0.23), (0.28)} and 5 require a **PP**($\overline{\overline{110000}}$) with its output attached to v_1 {(0.7), (0.11), (0.15),

TABLE II

Summary of the pulse sequences sent into loops $\mathbf{C}_1$ *and* $\mathbf{C}_2$

New Designation	Former Designation	Operation
		(I) Group (28′.a)–(28′.d) and (29′.a)–(29′.d) of Section 4.2.2
(0.1)*	(28′.a)	Inject $n + 1$ sequences $\overline{110110}$ at u_2 .
(0.2)	(28′.b)	Inject the sequence $\overline{11101001}$ at u_2 .
(0.3)*	(28′.c)	Inject $n + 1$ sequences $\overline{110000}$ at v_2 .
(0.4)	(28′.d)	Inject the sequence $\overline{11010}$ at v_2 .
(0.5)*	(29′.a)	Inject n sequences $\overline{110110}$ at u_1 .
(0.6)	(29′.b)	Inject the sequence $\overline{111101110110111001}$ at u_1 .
(0.7)*	(29′.c)	Inject n sequences $\overline{110000}$ at v_1 .
(0.8)	(29′.d)	Inject the sequence $\overline{110101101011001}$ at v_1 .
		(II) Group (29′.e′)–(29′.h′) of Section 4.2.4
(0.9)*	(29′.e′)	Inject n sequences $\overline{110110}$ at u_1 .
(0.10)	(29′.f′)	Inject the sequence $\overline{111101}$ at u_1 .
(0.11)*	(29′.g′)	Inject $n + 1$ sequences $\overline{110000}$ at v_1 .
(0.12)	(29′.h′)	Inject the sequence $\overline{1010}$ at v_1 .
		(III) Group (29′.e)–(29′.g) of Section 4.2.2
(0.13)*	(29′.e)	Inject $n + 1$ sequences $\overline{110110}$ at u_1 .
(0.14)	(29′.f)	Inject the sequence $\overline{1010}$ at u_1 .
(0.15)*	(29′.g)	Inject $n + 1$ sequences $\overline{110000}$ at v_1 .
		(IV) Group (30′.a)–(30′.g) and (31′.a)–(31′.c) of Section 4.2.3
(0.16)*	(30′.a)	Inject n sequences $\overline{110110}$ at u_2 .
(0.17)	(30′.b)	Inject the sequence $\overline{111101}$ at u_2 .
(0.18)*	(30′.c)	Inject n sequences $\overline{110000}$ at v_2 .
(0.19)	(30′.d)	Inject a single stimulus at v_2 .
(0.20)*	(30′.e)	Inject $n - 1$ sequences $\overline{110110}$ at u_2 .
(0.21)	(30′.f)	Inject the sequence $\overline{11010}$ at u_2 .
(0.22)*	(30′.g)	Inject $n - 1$ sequences $\overline{110000}$ at v_2 .
(0.23)*	(31′.a)	Inject n sequences $\overline{110110}$ at u_1 .
(0.24)	(31′.b)	Inject the sequence $\overline{11110111101}$ at u_1 .
(0.25)*	(31′.c)	Inject n sequences $\overline{110000}$ at v_1 .
		(V) Group (31′.d′) of Section 4.2.4
(0.26)	(31′.d′)	Inject the sequence $\overline{110101}$ at v_1 .
		(VI) Group (31′.d) of Section 4.2.3
(0.27)	(31′.d)	Inject the sequence $\overline{1101011010}$ at v_1 .
		(VII) Group (31′.e)–(31′.h) of Section 4.2.3
(0.28)*	(31′.e)	Inject n sequences $\overline{110110}$ at u_1 .
(0.29)	(31′.f)	Inject a single stimulus at u_1 .
(0.30)*	(31′.g)	Inject $n - 1$ sequences $\overline{110000}$ at v_1 .
(0.31)	(31′.h)	Inject the sequence $\overline{11010}$ at v_1 .

(0.25), (0.30)}. The delay requirements were expressed in Sections 4.2.2–4.2.4 by specifying the amount by which the delay from the turning on to the turning off of each **PP** is exceeded by the delay from the input a to the output b of its Φ. These excess amounts are restated in Table III, together with the Φ and **PP** data given above.

Regarding the 15 operations without an asterisk we need only specify which one of the inputs u_1, v_1 of $\mathbf{C}_1$ and u_2, v_2 of $\mathbf{C}_2$ each operation feeds. Actually, 5 feed into u_1 {(0.6), (0.10), (0.14), (0.24), (0.29)}, 5 feed into v_1 {(0.8), (0.12), (0.26), (0.27), (0.31)}, 3 feed into u_2 {(0.2), (0.17), (0.21)}, and 2 feed into v_2 {(0.4), (0.19)}. They are shown in a systematic arrangement in Table IV, together with the pertinent **P** data. We also show in this table a reference to an additional sequence that has to be injected at v_1 at a certain occasion. This is the entry (0.0), which refers to the sequence $\overline{10101}$ that occurs in the top line of Figure 38. Finally, we show (for later use) for each **P** its length and height. [There were several errors in von Neumann's

TABLE III

Summary of excess delay requirements for the periodic pulsers stimulating loops $\mathbf{C}_1$ *and* $\mathbf{C}_2$

Operation	PP Used	Output of PP Attached to	Φ Used	Excess Delay
				$3(\delta_2' + \delta_2'') + a_2$, where a_2 is
(0.5)				250
(0.9)				250
(0.13)	**PP**$(\overline{110110})$	u_1		244
(0.23)				238
(0.28)				238
			Φ_2	
(0.7)				250
(0.11)				244
(0.15)	**PP**$(\overline{110000})$	v_1		244
(0.25)				238
(0.30)				244
				$3(\delta_1' + \delta_1'') + a_1$, where a_1 is
(0.1)				241
(0.16)	**PP**$(\overline{110110})$	u_2		247
(0.20)				253
			Φ_1	
(0.3)				241
(0.18)	**PP**$(\overline{110000})$	v_2		247
(0.22)				253

TABLE IV

The pulsers used to stimulate loops $\mathbf{C}_1$ *and* $\mathbf{C}_2$

Operation	P Used	Output of P Attached to	Length ($2k$)	Height ($u + 2$)
(0.6)	P(1111011110110111001)		28	7
(0.10)	P(111101)		10	4
(0.14)	P(1010)	u_1	4	4
(0.24)	P(11110111101)		18	5
(0.29)	Single stimulus			1
(0.0)	P(10101)		6	5
(0.8)	P(110101101011001)		18	9
(0.12)	P(1010)	v_1	4	4
(0.26)	P(110101)		8	5
(0.27)	P(1101011010)		12	5
(0.31)	P(11010)		6	4
(0.2)	P(11101001)		10	5
(0.17)	P(111101)	u_2	10	4
(0.21)	P(11010)		6	4
(0.4)	P(11010)	v_2	6	4
(0.19)	Single stimulus			1

Table IV. These have been corrected according to the rule stated in the editorial summary at the end of Section 3.2.1.]

4.3.3 The read-write-erase unit **RWE.**[4] Figure 38 and the three tables in Section 4.3.2, together with the other listings given there, provide a definite basis for the construction announced at the beginning of that subsection. We can therefore approach this construction now in more complete and specific terms.

The operations that we wish to instrument primarily affect $\mathbf{C}_1$ and $\mathbf{C}_2$; i.e., they interact with the inputs u_1 , v_1 and u_2 , v_2 of these organs and with their outputs w_1 and w_2 . It is therefore best to start our constructions with the organs that are in direct contact with u_1 , v_1 , w_1 and u_2 , v_2 , w_2 . These are specified in Tables III and IV, together with the Ψ referred to in lines 2–4 of Figure 38. The functioning of all these organs is controlled in the manner described in Figure 38 and in Table II.

[4] [Von Neumann's title for the present subsection was "The effector organs of **B**. Positioning and connecting these organs."]

Hence we begin by providing and positioning the organs of Tables III and IV, as well as Ψ.

Table IV calls for 16 organs, 14 of which are pulsers (**P**), while 2 are merely providers of single stimuli. Each of these is actuated by a single input, coming from **RWEC** (cf. Fig. 37), and has a single output, feeding into one of u_1, v_1, u_2, v_2.

Table III calls for the following organs. First, 2 periodic pulsers {**PP**$(\overline{110110})$ and **PP**$(\overline{110000})$} are required, but since each of these is required with two different output connections (u_1 and v_1 in the first case, u_2 and v_2 in the second case), therefore only 4 periodic pulsers are actually needed. (Table III also shows that each one of these **PP** is required in several different situations—their numbers are 5, 5, 3, 3, respectively—requiring different "excess delays." This might incline one to introduce a separate **PP** for each such case, i.e., 16 in all, instead of the above 4. However, it is more convenient to take care of this by other arrangements, which will appear further below, and to introduce only 4 **PP**'s, as indicated above.) Second, Table III requires a triple-return counter (Φ), but since this is required with two different c, d-connections (v_1, w_1 in one case and v_2, w_2 in the other) actually two such organs are called for. Each **PP** is controlled by two inputs a_+ and a_-, coming from **MC**, and has a single output feeding into one of u_1, v_1, u_2, v_2. Each Φ has an input-output pair a, b connected to **MC**, and an input-output pair c, d connected to v_1, w_1 or to v_2, w_2.

Finally, we need the $\overline{1}$ vs. $\overline{10101}$ discriminator Ψ. This has a single input a which is attached to w_1 and two outputs b, c going to **RWEC** (cf. Fig. 38).

[Von Neumann overlooked one point when attaching the input a of Ψ to the output w_1 of $\mathbf{C}_1$. When $\mathbf{C}_1$ is used to time the lengthening (or shortening) of the timing loop $\mathbf{C}_2$, single pulses will go from w_1 to the triple-return counter Φ_1 in Figure 39. But these pulses will also enter Ψ, which will then indicate a $\overline{1}$ received to the constructing unit **CU**.

We will assume that the constructing unit **CU** will be built so as to ignore these spurious signals. Alternatively, the input to Ψ could be closed by a gate controlled from a **PP**$(\overline{\overline{1}})$ when Φ_1 is in use. This would, of course, change the design of **RWE** somewhat.]

All these organs are shown in Figure 39 in their relative positions with respect to $\mathbf{C}_1$, $\mathbf{C}_2$ (i.e., u_1, v_1, w_1, u_2, v_2, w_2) on one hand and with respect to **MC** on the other. The sub-organs contained in this assembly are **P**'s (designated 0.6–0.24, 0.0–0.31, 0.2–0.21, 0.4), **PP**'s, Φ's (designated Φ_1, Φ_2), and a Ψ. All of these are not shown in their true sizes. The entire assembly is part of **MC**. To the left, across the

vertical dashed line, it is attached to the remainder of **MC**. To the right it is attached to $\mathbf{C}_1$, $\mathbf{C}_2$, as indicated. The **P** of Table IV are indicated by their 0 symbols, except for the two single-stimulation organs which are shown as direct channels. All inputs and outputs of the sub-organs that occur (**P**, **PP**, Φ, Ψ) are properly indicated. The various inputs and outputs that connect this assembly with the remainder of **MC** are marked with symbols that are self-explanatory. Inspection shows that there are 30 connections of the latter kind, of which 4 are inputs to **MC** and 26 are outputs from **MC**.

Note, furthermore, that while the **P**, **PP**, and Φ are in their standard orientations, i.e., in the orientations of Figures 16, 18, and 23, the Ψ is reflected about the vertical in comparison to its standard arrangement in Figure 25. Clearly, this calls merely for trivial transformations of the construction. (Fig. 25 contains a **P** and a **D**, and these, too, must be reflected about the vertical. This calls for corresponding transformations in connection with Figs. 16 and 22. These, too, are harmless; cf. the discussion in the middle part of Sec. 3.6.1 in connection with Fig. 29.)

The assembly of Figure 39 must be controlled, as we pointed out at the beginning of this section, according to the scheme described in Figure 38 and in Table II. The organs which effect this control must occupy the deeper interior of **MC**. We saw in Figure 39 that the number of connections between this region, and between the assembly that was explicitly described, is quite large, 4 inputs and 26 outputs. This makes it a practical certainty that the lines representing these 30 connections will have to cross each other many times in order to find their proper endings on the control organs within **MC**. This means that the need for the organ that circumvents the difficulties of line-crossing has arisen; i.e., we need the coded channel of Section 3.6.

We are not yet in a position to lay out the entire coded channel that will be required, i.e., to select the appropriate arrangement, as discussed in the first part of Section 3.6.1 and schematically shown in Figures 28g–28k. This will be done later. For the time being, we will concern ourselves only with the portion that is adjacent to the assembly of Figure 39, i.e., that may be thought of as extending along the vertical dotted line on the left edge of that figure.

We saw above that 30 lines (4 inputs and 26 outputs) connect with this portion. Each of these is stimulated (or stimulates) independently of the others. That is, in the notations used at the beginning of Section 3.6.1, they correspond to inputs a_ν or outputs b_ν (from the point of view of the coded channel) with different ν. Hence the total range

$\nu = 1, \cdots, n$ must allow for at least 30 different possibilities. That is, necessarily $n \geqq 30$. Actually it is advisable to choose n somewhat larger, since further requirements of the control **MC** will call for additional ν values for the inputs a_ν and outputs b_ν of the coded channel. (Cf. below.)

On this basis we will now choose the m and k referred to in the middle part of Section 3.6.1, which determine the coded sequences $\overline{i_\nu^1 \cdots i_\nu^m}$ that correspond to the ν ($\nu = 1, \cdots, n$). In this respect equation (12′) and the observation following equation (13′) are relevant: given m, these observations determine k ($= (m+1)/2$ or $m/2$, whichever is an integer) and the former determines

$$\text{Max } n = \binom{m-1}{k-1}.$$

The following m are critical:

m	7	8	9
k	4	4	5
Max n	20	35	70

Since we want $n \geqq 30$, with a reasonable margin to spare, $m = 7$ is inadequate, $m = 8$ is marginal (it will prove to be inadequate, cf. below), while $m = 9$ is presumably adequate (it will prove to be adequate; cf. below). We choose therefore

$$(32') \qquad m = 9, k = 5$$

so that n is limited by

$$(33') \qquad n \leqq 70$$

only.

According to Section 3.6.1, each sequence $\overline{i_\nu^1 \cdots i_\nu^m}$ begins with a $\overline{1}$, has length $m = 9$, and contains $k = 5$ $\overline{1}$'s. That is, it is $\overline{1\ i_\nu^2 \cdots i_\nu^9}$, where among the $i^2, \cdots, i^9$ there are four 1's and four 0's. We know that there are precisely

$$\binom{m-1}{k-1} = \binom{8}{4} = 70$$

such sequences. Let us order them lexicographically, and use the $\nu = 1, \cdots, 70$, to enumerate them in this order. This, then, defines $\overline{i_\nu^1 \cdots i_\nu^m} = \overline{1\ i_\nu^2 \cdots i_\nu^9}$ for each $\nu = 1, \cdots, 70$, thus certainly including the $\nu = 1, \cdots, n$ {for any n according to condition (33′)}.

Let us assign to the 30 a_ν and b_ν of Figure 39 (i.e., to the $u_1 \cdot 1, \cdots,$

$\Phi_2 \cdot a$ occurring there) the numbers $\nu = 1, \cdots, 30$, in a vertically descending sequence, according to that figure.

[Von Neumann proceeded to calculate the dimensions of the assembly of Fig. 39.

It turns out that the height of CC_1 is greater than the height of **RWE**. Since von Neumann wanted to place each pulser and decoder of CC_1 nearly opposite the organ of **RWE** to which it is connected, the height of CC_1 controls the height of the complex CC_1-**RWE**. Now CC_1 has 4 pulsers **P** and 26 decoders **D**. Von Neumann said that the height of each is 7, and, allowing a protective strip of **U**'s between each two consecutive organs, he arrived at a height of 239 for CC_1.

But this is wrong. The sequences von Neumann was using in his coded channel are of length nine; each begins with a one and contains four additional ones. By the rule of Section 3.3 for computing the size of a decoder, the height of a decoder for a sequence of length nine with five ones is either 10 or 11, depending on the specific sequence. In fact, the rule of Section 3.3 often gives a height which is greater than the actual height of a decoder constructed by the design algorithm of Section 3.3. But even when this is taken into account, a height of over 300 units is needed by the CC_1 of von Neumann's design. While this is a mere error of calculation, it upsets most of von Neumann's later calculations concerning the size of **MC** and the delays through **MC**.

This error cannot be corrected simply by increasing the height of CC_1 and thereby the height of **MC**. The reason it cannot be so corrected concerns the proper phasing of the periodic pulsers **PP** and triple-return counters Φ of **RWE**. Each **PP** should be on for about three times the delay around the loop C_1 or the loop C_2 (see Table II of Sec. 4.3.2). But there is a delay within each Φ and between the secondary input-output of each Φ and its connecting loop. Moreover, the **PP** of **RWE** are controlled from **RWEC**. Hence a primary output from a Φ of **RWE** must go through CC_1, the main channel, CC_2, **RWEC**, the main channel again, and CC_1 again, all before it can turn off the **PP** of **RWE**. Altogether, about 2000 time units are lost in this way. Thus, for correct phasing, the starting of a **PP** of **RWE** must be delayed about 2000 time units over the starting of its associated triple-return counter Φ. Some of this delay is obtained in the control organs **CO** of **RWEC**, but most of it is obtained in area **W**. It will turn out that area **W** is not large enough to give the delay von Neumann needed for his design.

Von Neumann's error can be corrected by extending area **W** to the right. This destroys the rectangular shape of the memory control

MC, however, and there are a number of ways of correcting the error without changing the size of **MC**. In Chapter 5 we will redesign units **Z** and **W** so that their heights can be reduced enough to allow $\mathbf{CC}_1$ to extend into the area now occupied by **Z**.

Von Neumann calculated the width of **RWE** and of $\mathbf{CC}_1$ as follows. The widest organ in **RWE** is the pulser labeled "0.6," which produces a sequence with 14 ones (see Table II of Sec. 4.3.2). This calls for a width of 28. All other organs of **RWE** are of width 24 or less. Von Neumann added 1 unit for the vertical channel from this pulser to input u_1 of loop $\mathbf{C}_1$, obtaining a total width of 29 units for **RWE**. The coded sequences entering and leaving $\mathbf{CC}_1$ contain 5 ones each; hence the pulsers and decoders of $\mathbf{CC}_1$ are each of width 10. Von Neumann added a strip on each side in accordance with the design of Figure 29. Hence $\mathbf{CC}_1$ is 12 units wide. The combined width of $\mathbf{CC}_1$-**RWE** is thus 41 units.

Actually there is a small error here too. The vertical channel from the output of the pulser labeled "0.6" would pass by, and receive stimuli from, the confluent state in the lower right-hand corner of this pulser, so that the wrong sequence would enter input u_1 of loop $\mathbf{C}_1$. There are a number of ways in which this error can be corrected so that the complex $\mathbf{CC}_1$-**RWE** can be accommodated within a width of 41. The pulser can be turned upside down. The pulser can be redesigned so as to be narrower and higher. The best way is to redesign the coded channel, and hence $\mathbf{CC}_1$, as follows.

Let us count the number of pairwise distinct coded sequences needed for the coded channel of **MC**. The outputs o_2, o_3, o_5 of **CU** and the input i_3 of **CU** pass through area X and require 4 coded sequences. It will turn out that 4 coded sequences are needed in area **Z**. An examination of Figures 39 and 41 shows that 31 additional coded sequences are needed. Hence 39 pairwise distinct coded sequences are needed for the coded channel of **MC**.

Recall that m is the length of the coded sequence and k is the number of ones it contains. As von Neumann correctly calculated above, his algorithm calls for $m = 9$ and $k = 5$ when 39 sequences are needed. But a better choice is $m = 9$ and $k = 4$; this gives 56 different sequences, which is more than enough. The pulsers and decoders for $k = 4$ are of width 8, a saving of two units over von Neumann's pulsers and decoders.

Von Neumann later used the width of 12 for $\mathbf{CC}_2$ and $\mathbf{CC}_3$ and calculated a width of 18 for **RWEC**. He calculated a height of 545 for $\mathbf{CC}_3$; lesser heights are required for $\mathbf{CC}_2$ and **RWEC**. Thus he obtained a width of 42 and a height of 545 for the complex $\mathbf{CC}_2$-

RWEC-CC$_3$. But for this height of 545 to obtain, the decoders of **CC$_2$** cannot always be placed so that their outputs feed directly into the organs of **RWEC**, and the pulsers of **CC$_3$** cannot always be placed so that their inputs come directly from the organs of **RWEC**. Vertical channels are needed for their connections. These vertical channels can be provided within the limits of von Neumann's dimensions for the complex **CC$_2$-RWEC-CC$_3$** by using decoders in **CC$_2$** and pulsers in **CC$_3$** which are of width 8, so that **CC$_2$** and **CC$_3$** can each be of width 10 and **RWEC** can be of width 22.

Hence we modify von Neumann's design of the coded channel by choosing $m = 9$ and $k = 4$, i.e., coded sequences of length 9 containing 4 ones. By the rule of Section 3.2.1 for computing the size of a pulser, the pulsers of this coded channel will be of width 8 and height 7. By the rule of Section 3.3 for computing the size of a decoder, the decoders of this coded channel will be of width 8 and height 11. But it turns out that every decoder constructed by the algorithm of Section 3.3 has a height of 10. This means that **CC$_1$** can be accommodated within a height of 320 and a width of 10, including insulating strips of unexcitable states.

We thus end up with the following dimensions for the memory control **MC** and its main parts (cf. Fig. 37):

MC: 547 cells high, 87 cells wide
RWE: 320 cells high, 31 cells wide
RWEC: 545 cells high, 22 cells wide
CC$_1$: 320 cells high, 10 cells wide
CC$_2$: 545 cells high, 10 cells wide
CC$_3$: 545 cells high, 10 cells wide.

These figures presuppose that the design can be completed without enlarging any of the organs of **RWE**. This presupposition will be confirmed in Chapter 5.]

4.3.4 The basic control organ **CO** *in* **MC**. We have completed the first task formulated at the beginning of Section 4.3.3: providing and positioning the organs of Tables III and IV in Section 4.3.2, and of Ψ, i.e., of those organs that are in direct contact with u_1, v_1, w_1, u_2, v_2, w_2 (i.e., with **C$_1$** and **C$_2$**). We can therefore now pass to the second task formulated there: controlling the organs referred to above, in the manner described in Figure 38 and in Table II of Section 4.3.2.

It is advantageous to deal first with a certain preliminary problem, before coming to grips with the above task in its entirety. This preliminary problem is that of controlling the functioning of the four **PP** of Figure 39 according to the requirements formulated in Table III in Section 4.3.2, and in the discussion immediately preceding it.

According to these, each **PP** must first be turned on and then turned off, so that the delay between these two events exceeds by a specified amount the delay from the stimulus at the input a of a certain Φ to the response at its output b. Each **PP** is attached to one of u_1, v_1, u_2, v_2. Let us designate the one of these under consideration by $\bar{u}$. Let us also designate its Φ (Φ_1 or Φ_2) by $\bar{\Phi}$. Then we have to stimulate, according to the above, first $\bar{u}\cdot a_+$ and $\bar{\Phi}\cdot a$, and then let the response at $\bar{\Phi}\cdot b$ stimulate $\bar{u}\cdot a_-$. The delays of these processes must, in addition, be so adjusted that they produce together the desired excess delay (of $\bar{u}\cdot a_+$ to $\bar{u}\cdot a_-$ over $\bar{\Phi}\cdot a$ to $\bar{\Phi}\cdot b$) referred to above.

This control organ has the symbol **CO**.

[See Figure 40, which differs from von Neumann's design in four respects. First, von Neumann placed output b above output c; our positioning is better, since the stimulus from b needs to be delayed relative to the stimulus from c. Second, von Neumann omitted the insulating row of **U**'s under the **PP** ($\bar{\bar{1}}$); this is needed since the bottom row of a **PP** ($\bar{\bar{1}}$) contains some confluent states. Third, von Neumann's **CO** had a width of 17, while the **CO** of Figure 40 has a width of 16. Fourth, and most important, von Neumann used his **PP** ($\bar{\bar{1}}$) of Figure 17b. This is wrong for the following reason.

At most one control organ **CO** of **RWEC** (Fig. 41) is on at any time; when it is on it has control of **RWE**. Now the output from the primary output of a triple-return counter (Φ_1 or Φ_2) of **RWE** (Fig. 39) causes stimuli to enter the stop inputs a_- of several **CO** of **RWEC**. As mentioned at the end of Section 3.2.2, if von Neumann's **PP** ($\bar{\bar{1}}$) of Figure 17 is stimulated at the stop input a_- when it is inactive, it will be damaged. The alternate periodic pulser **PP** ($\bar{\bar{1}}$) of Figure 20 is not harmed under these circumstances, and therefore we have used it in **CO**.]

There are 2 different **PP**, but they are attached to 4 different $\bar{u}$ ($\bar{u}$ determined the **PP** as well as its $\bar{\Phi}$; cf. Table III); hence we might expect that we will need 4 organs **CO**. However, these 4 cases are further subdivided into 16 sub-cases, according to Tables II and III, each sub-case corresponding to a 0 with an asterisk in these tables. Indeed, each formula (0.x) with an asterisk requires a different follow-up move and sequence of moves according to Table II, and a different excess delay according to Table III. (Those lines of Table III, which happen to call for the same excess delay, differ in their $\bar{u}$.) Consequently we need 16 organs **CO**, one for each (0.x) with an asterisk.

Consider now a specific **CO**, i.e., a specific (0.x) with an asterisk. The input a is the primary input. A stimulus at a should go to $\bar{u}\cdot a_+$

and to $\bar{\Phi}\cdot a$; these will therefore be connected to two outputs, to b and to c, respectively. Note that the $\bar{u}\cdot a_+$ and the $\bar{\Phi}\cdot a$ referred to must be thought of as certain suitable inputs of the coded channel, while the $\bar{u}\cdot a_+$ and $\bar{\Phi}\cdot a$ in Figure 39, which are the ones ultimately aimed at, are outputs of the coded channel. Hence the functioning of this organ, involving $\bar{u}\cdot a_+$ and $\bar{\Phi}\cdot a$, will be affected by the corresponding delays due to the use of the coded channel. This is equally true for the uses involving $\bar{u}\cdot a_-$ and $\bar{\Phi}\cdot b$, which will appear further below. All these coded channel delays will affect, additively or subtractively, the excess delays prescribed in Table III. Hence precise delay adjustments will have to be made in **CO** to account for these things, and it will be possible to make a final determination of these only after having effected the precise positioning of this **CO** with respect to the coded channel, and the laying out and positioning of the entire coded channel with respect to **RWE**. We must therefore, for the time being, leave these delay adjustments in a schematic and adjustable condition. We do this by assigning an area **D**, of as yet unspecified dimensions, to the delay path from a to b which will become necessary for this purpose; this delay path is indicated by the symbol ⓓ.

We can now go on to the other functions of **CO**. These are the following. A response stimulus from $\bar{\Phi}\cdot b$ must stimulate $\bar{u}\cdot a_-$, and also the follow-up move that Table II (and, more broadly, the general scheme of Fig. 38) prescribes for this **CO** case {i.e., for this 0 with an asterisk}.

It would not do, however, to take a connection directly from $\bar{\Phi}\cdot b$ to $\bar{u}\cdot a_-$ and to the input of the follow-up move. Indeed, the same $\bar{\Phi}\cdot b$ corresponds to several (6 for Φ_1 and 10 for Φ_2; cf. Table III) of our **CO** cases. Accordingly, it corresponds to two possible $\bar{u}$'s (u_2, v_2 for Φ_1 and u_1, v_1 for Φ_2; cf. Table III) and to several possible follow-up moves (6 or 10, respectively; cf. above). Hence our **CO** must be provided with a memory, to cause the $\bar{\Phi}\cdot b$ stimulus to activate only its (the **CO**'s) own $\bar{u}\cdot a_-$ and follow-up move.

Actually, this is not necessary for the $\bar{u}\cdot a_-$; i.e., $\bar{\Phi}\cdot b$ could be tied to all of $u_1\cdot a_-$, $v_1\cdot a_-$, $u_2\cdot a_-$, $v_2\cdot a_-$. This would merely turn off, in addition to the one **PP** which has been previously turned on (by $\bar{u}\cdot a_+$) and which it is intended to turn off now (because it corresponds to the correct $\bar{u}$), also the three other **PP** which had not been previously turned on at all. Turning off a **PP** [of Fig. 39] which has not been turned on has no effects whatever [cf. the end of Sec. 3.2.2.]

For the follow-up move, on the other hand, this special control of the $\bar{\Phi}\cdot b$ stimulus is necessary. Indeed, if this stimulus were per-

mitted to start all the follow-up moves corresponding to its $\bar{\Phi}$ (6 or 10, respectively; cf. above) together, this would corrupt the functioning of the automaton altogether. Since the special control of the $\bar{\Phi} \cdot b$ stimulus is needed for its effects on the follow-up moves, it is simplest to apply that control to the effects on $\bar{u} \cdot a_-$ also. We will do this in what follows.

This special control (i.e., the coincidence of the fact that the specific **CO** under consideration is the one that has been activated, with the response of $\bar{\Phi} \cdot b$) requires a memory organ, as noted above, and a subsequent coincidence organ. The memory organ is, of course, a **PP**$(\bar{1})$, turned on from a of **CO**. The **PP**$(\bar{1})$ then supplies continuously one stimulus to the coincidence organ, the **C** in the lower right corner of Figure 40. When the $\bar{\Phi} \cdot b$ response arrives (i.e., when the input d is stimulated), then the coincidence organ receives its other stimulus. The response of the coincidence organ then emits stimuli at the outputs e and f. Output e is connected to $\bar{u} \cdot a_-$, while output f is connected to the follow-up move.

The input d also clears the memory, i.e., turns the **PP**$(\bar{1})$ off. [An examination of Figs. 40 and 20 shows that] the turning off of **PP**$(\bar{1})$ comes too late to interfere with the coincidence-sensing operation.

The delay area **D** is now allotted a width of 15, while its height h is kept adjustable. We will choose h later, when the detailed laying-out and adjusting, referred to above, will take place. The discussion and determination of the delays within **CO** must, of course, also be postponed until then.

4.3.5 The read-write-erase control **RWEC**.[5] [See Fig. 41.] Let us now consider the general problem outlined at the beginning of Section 4.3.4: controlling the organs that are in direct contact with the connecting loop $\mathbf{C}_1$ and the timing loop $\mathbf{C}_2$ (i.e., $\mathbf{CC}_1$ plus **RWE** of Fig. 37), according to Figure 38 and Table II in Section 4.3.2. This will, of course, be based on the use of the control organ **CO** of Section 4.3.4.

Figure 38 shows that there is still a memory requirement to be satisfied: the instructions "starts α_0" (line 5) and "starts α_1" (line 6) amount to this.

Activities depending on one or the other of these two instructions (marked "α_0 active" or "α_1 active," in lines 9 and 15) occur twice for each. It is simplest to attach a "local" memory with this function at each one of the four points referred to. Hence we will have to provide four **PP**$(\bar{1})$'s for this purpose. Thus the activities referred to above,

[5] [Von Neumann's title was "The control-area in **B**. Overall dimensions of **B**."]

which depend on α_0 or α_1 being "active" (lines 9 and 15) will be derived from a coincidence organ (a **C**), which gets one stimulus from the relevant **PP**, and the other stimulus from the logical antecedent of the activity in question, according to Figure 38.

Finally, there exist two orders "stops α_0" and two orders "stops α_1" (whichever happened to be active; cf. lines 10 and 16); i.e., at this point the two **PP** of α_0 or of α_1, respectively, must be turned off. However, here a deviation from Figure 38 is permissible. Indeed, these "stop," (i.e., turn-off) operations can be delayed to the end of these sequences {i.e., to the points where i_3 is stimulated (lines 12 and 19)}. Here all four sequences meet. Therefore, here all these turn-offs can be effected by a single stimulus, which stops all four **PP**'s. In this way not only the two **PP** of that one of α_0, α_1 which was turned on will now be turned off, but also those of the other one, which was not turned on at all. The latter measure, however, is harmless; it has no effects whatever. [As explained at the end of Section 3.2.2, this is so for von Neumann's general periodic pulser **PP**, but not for his **PP**$(\bar{\bar{1}})$ of Figure 17b. Therefore the alternate **PP**$(\bar{\bar{1}})$ of Figure 20 should be used here. That is, the four **PP**$(\bar{\bar{1}})$ of Figures 41b, 41c, and 41e should be the alternate **PP**$(\bar{\bar{1}})$ of Figure 20.]

The stimulus to be used for this purpose is obviously i_3 itself.

Based on all these considerations, we arrive at the assembly shown in Figure 41.

The structure of this figure is very similar to that of Figure 39. The vertical dotted lines which border it on both sides have the same role as the single vertical dotted line in Figure 39. As discussed in Section 4.3.3, the latter indicated the portion of the coded channel adjacent to the assembly of Figure 39 (i.e., $\mathbf{CC}_1$, which is adjacent to **RWE** in Fig. 37). Similarly now the two first mentioned lines indicate the portions of the coded channel adjacent to the two edges of our assembly in Figure 41. We will discuss the relationship of these three portions of the coded channel ($\mathbf{CC}_1$, $\mathbf{CC}_2$, and $\mathbf{CC}_3$) further below; it will appear that they determine together most of the coded channel of the memory control **MC**.

All inputs and outputs of the sub-organs that occur in Figure 41 (**PP**, **CO**) are properly indicated. The various inputs and outputs that connect this assembly with the coded channel (i.e., with $\mathbf{CC}_2$ and $\mathbf{CC}_3$) and with the constructing unit **CU** are marked with symbols that are self-explanatory.

Everything on the $\mathbf{CC}_2$ side of **RWEC** is an output of $\mathbf{CC}_2$; inspection shows that there are 33 of these. Repeatedly, there are several specimens of the same output b_ν in positions that are immediately

TABLE V

How the control organs of **RWEC** *control the pulsers and periodic pulsers of* **RWE**

Operation	Control Organ	Outputs of **RWEC**, Inputs of **RWE**	Triple-return Counter Used	Sequences into Connecting Loop C_1 or into Connecting Loop C_2	Effect on C_1 or C_2
Lengthening C_2 and C_1					
Lengthen C_2	CO_1	$u_2 \cdot a_+$, $u_2 \cdot a_-$	Φ_1	Inject $n+1$ sequences $\overline{110110}$ at u_2.	Figs. 32a–32b
		$u_2 \cdot 1$		Inject 11101001 at u_2.	Figs. 32b–32c
	CO_2	$v_2 \cdot a_+$, $v_2 \cdot a_-$	Φ_1	Inject $n+1$ sequences $\overline{110000}$ at v_2.	Figs. 32c–32d
		$v_2 \cdot 1$		Inject 11010 at v_2.	Figs. 32d–32e
Lengthen bottom of C_1	CO_3	$u_1 \cdot a_+$, $u_1 \cdot a_-$	Φ_2	Inject n sequences $\overline{110110}$ at u_1.	Figs. 33a–33b
		$u_1 \cdot 1$		Inject 1111011110110111001 at u_1.	Figs. 33b–33c
	CO_4	$v_1 \cdot a_+$, $v_1 \cdot a_-$	Φ_2	Inject n sequences $\overline{110000}$ at v_1.	Figs. 33c–33d
		$v_1 \cdot 2$		Inject 110101101011001 at v_1.	Figs. 33d–33e
Lengthen top of C_1 and write "zero" in x_n	CO_5	$u_1 \cdot a_+$, $u_1 \cdot a_-$	Φ_2	Inject n sequences $\overline{110110}$ at u_1.	Figs. 33e–36a
		$u_1 \cdot 2$		Inject 111101 at u_1.	Figs. 36a–36b
	CO_6	$v_1 \cdot a_+$, $v_1 \cdot a_-$	Φ_2	Inject $n+1$ sequences $\overline{110000}$ at v_1.	Figs. 36b–36c
		$v_1 \cdot 3$		Inject 1010 at v_1.	Figs. 36c–36d
Lengthen top of C_1 and write "one" in x_n	CO_7	$u_1 \cdot a_+$, $u_1 \cdot a_-$	Φ_2	Inject $n+1$ sequences $\overline{110110}$ at u_1.	Figs. 33e–33f
		$u_1 \cdot 3$		Inject 1010 at u_1.	Figs. 33f–33g
	CO_8	$v_1 \cdot a_+$, $v_1 \cdot a_-$	Φ_2	Inject $n+1$ sequences $\overline{110000}$ at v_1.	Figs. 33g–33h

Shortening $\mathbf{C}_2$ and $\mathbf{C}_1$					
Shorten $\mathbf{C}_2$	$\mathbf{CO}_9$	$u_2 \cdot a_+$, $u_2 \cdot a_-$	Φ_1	Inject n sequences $\overline{110110}$ at u_2.	Figs. 32a–34a
		$u_2 \cdot 2$		Inject $\underline{111101}$ at u_2.	Figs. 34a–34b
	$\mathbf{CO}_{10}$	$v_2 \cdot a_+$, $v_2 \cdot a_-$	Φ_1	Inject n sequences $\underline{110000}$ at v_2.	Figs. 34b–34c
		$v_2 \cdot 2$		Inject a single stimulus at v_2.	Figs. 34c–34d
	$\mathbf{CO}_{11}$	$u_2 \cdot a_+$, $u_2 \cdot a_-$	Φ_1	Inject $n-1$ sequences $\underline{110110}$ at u_2.	Figs. 34d–34e
		$u_2 \cdot 3$		Inject $\underline{11010}$ at u_2.	Figs. 34e–34f
	$\mathbf{CO}_{12}$	$v_2 \cdot a_+$, $v_2 \cdot a_-$	Φ_1	Inject $n-1$ sequences $\overline{110000}$ at v_2.	Figs. 34f–34g
Shorten bottom of $\mathbf{C}_1$ and write in cell x_n	$\mathbf{CO}_{13}$	$u_1 \cdot a_+$, $u_1 \cdot a_-$	Φ_2	Inject n sequences $\overline{110110}$ at u_1.	Figs. 33a–35a
		$u_1 \cdot 4$		Inject $\underline{11110111101}$ at u_1.	Figs. 35a–35b
	$\mathbf{CO}_{14}$	$v_1 \cdot a_+$, $v_1 \cdot a_-$	Φ_2	Inject n sequences $\underline{110000}$ at v_1.	Figs. 35b–35c
	Write "zero"	$v_1 \cdot 4$		Inject $\underline{110101}$ at v_1.	Figs. 35c–35d with **U** at x_n
	Write "one"	$v_1 \cdot 5$		Inject $\overline{1101011010}$ at v_1.	Figs. 35c–35d
Shorten top of $\mathbf{C}_1$	$\mathbf{CO}_{15}$	$u_1 \cdot a_+$, $u_1 \cdot a_-$	Φ_2	Inject n sequences $\overline{110110}$ at u_1.	Figs. 35d–35e
		$u_1 \cdot 5$		Inject a single stimulus at u_1.	Figs. 35e–35f
	$\mathbf{CO}_{16}$	$v_1 \cdot a_+$, $v_1 \cdot a_-$	Φ_2	Inject $n-1$ sequences $\underline{110000}$ at v_1.	Figs. 35f–35g
		$v_1 \cdot 6$		Inject $\underline{11010}$ at v_1.	Figs. 35g–35h

(See also Figures 39 and 41.)

adjacent to each other. (In one case two neighboring $\Phi_1 \cdot b$, in one case four neighboring $\Phi_1 \cdot b$, and in five cases two neighboring $\Phi_2 \cdot b$.) In each one of these cases it would have been possible to replace the complex of neighboring b_ν's by a single b_ν, thus shortening (i.e., reducing the height of) the $\mathbf{CC}_2$ side. However, the height of the $\mathbf{CC}_2$ side is even so not controlling (cf. further below); hence there is no advantage in reducing it. On the other hand, a merger of b_ν's would necessitate the introduction of vertical distribution channels in the assembly to the right of $\mathbf{CC}_2$ (leading to the various sub-organs **CO** that these b_ν, i.e., these $\Phi_1 \cdot b$ and $\Phi_2 \cdot b$, feed), and thereby the width of the assembly would be effectively increased.

Everything on the $\mathbf{CC}_3$ side is an input of $\mathbf{CC}_3$; inspection shows that there are 68 of these.

In addition to these there are two inputs, o_1 and o_4, and two outputs, i_1 and i_2, on the top side of Figure 41.

The portions $\mathbf{CC}_2$ and $\mathbf{CC}_3$ of the coded channel that we have now introduced have as inputs a_ν and outputs b_ν (always from the point of view of the coded channel, i.e., of $\mathbf{CC}_2$ or $\mathbf{CC}_3$) all those that occurred on $\mathbf{CC}_1$ (in Fig. 39), and in addition the following new ones: β, i_3 (these occur as both inputs and outputs), o_2, o_3, o_5 (these occur as outputs only). Thus the 30 distinct ν that were required for $\mathbf{CC}_1$ (cf. Sec. 4.3.3) are augmented by another 5. This ties into our discussion of formulas (32′) and (33′). It means that we have so far $n \geqq 35$. (There will be yet another increase of n; cf. later.)

[In the remainder of this section von Neumann calculated the dimensions of $\mathbf{CC}_2$, **RWEC**, and $\mathbf{CC}_3$; see the end of Section 4.3.3 for a discussion of his results.

We add Table V, which collates the information in von Neumann's Figures 32–36, 39, and 41 and Tables II–IV. Table V shows how the control organs of **RWEC** control the pulsers and periodic pulsers of **RWE**.]

AUTOMATA SELF-REPRODUCTION

5.1 Completion of the Memory Control MC

[*5.1.1 The rest of the manuscript.* Von Neumann's manuscript continues for six further sections and then abruptly terminates. These sections are devoted mainly to detailed calculations of the delays within the memory control **MC**. Most of these delay calculations are wrong, owing to the errors von Neumann made in designing and calculating the sizes of various organs and units in Chapter 3. For this reason we will not reproduce the balance of the manuscript but will summarize it instead. The omitted portion of the manuscript is of about the length of Chapter 3 and contains 7 figures.

The organization and operation of the memory control **MC** is summarized in Section 4.3.1 and Figure 37. The status of von Neumann's design of **MC** at the end of Chapter 4 is as follows.

Read-write-erase unit **RWE:** The pulsers, periodic pulsers, triple-return counters, and discriminator that constitute **RWE** have been designed and their order in **RWE** has been decided; see Figure 39. The exact positions of these organs in **RWE** have not been decided upon, but each is to be so placed that it is fed nearly directly by a decoder of $\mathbf{CC}_1$ and/or it feeds a pulser of $\mathbf{CC}_1$ nearly directly. **RWE** is to be 320 cells high and 31 cells wide. There is a wide range of choice for the position of exit w_2 of the timing loop $\mathbf{C}_2$. Von Neumann placed it 48 cells above the bottom of **RWE**.

Read-write-erase-control unit **RWEC: RWEC** is mainly composed of control organs **CO**; these have been designed except for specifying the size of delay area **D** and the design of the delay path ℬ (Fig. 40). The 16 **CO**'s and the 4 $\mathbf{PP}(\bar{1})$'s are arranged as in Figure 41. The exact positions of these organs in **RWEC** have not been decided upon, but each organ is to be placed so as to satisfy as nearly as possible the principle that a decoder of $\mathbf{CC}_2$ feeds it directly and it feeds a pulser of $\mathbf{CC}_3$ directly. **RWEC** is to be 545 cells high and 22 cells wide.

Coded channel: **RWEC** and **RWE** are to be positioned as in Figure

37. The coded channel consists of $\mathbf{CC_3}$, **X**, **Z**, $\mathbf{CC_1}$, $\mathbf{CC_2}$, and the main channel. The main channel extends from the solid dot near the bottom middle of Figure 37 to the solid square in the upper left-hand corner of the same figure—more specifically, the main channel ends where it feeds the topmost pulser of $\mathbf{CC_2}$. The code to be used in the coded channel has the following characteristics. Each coded sequence will begin with a one, will contain exactly three other ones, and will be of length 9 or less. This code allows for 56 different sequences; it will be confirmed later that less than this number of sequences is required. These sequences have not yet been assigned to the inputs and outputs of the coded channel. The pulsers of the coded channel will be of width 8 and height 7, and the decoders of width 8 and height 10. $\mathbf{CC_1}$ is to be of width 10 and height 320. $\mathbf{CC_2}$ and $\mathbf{CC_3}$ are each to be of width 10 and height 545.

The memory control **MC** will be 547 cells high and 87 cells wide, assuming that the undesigned organs and units can be accommodated in this space.

We will now summarize what von Neumann accomplished in the part of the manuscript we are not publishing. He first combined the read-write-erase unit **RWE** and its control **RWEC** as in Figure 37. An examination of Section 4.3.3 and Figure 39 shows that information flows in both directions through $\mathbf{CC_1}$, so that $\mathbf{CC_1}$ contains both decoders and pulsers. An examination of Section 4.3.5 and Figure 41 shows that **RWEC** receives information only from $\mathbf{CC_2}$ and transmits information only to $\mathbf{CC_3}$. Consequently, $\mathbf{CC_2}$ contains only decoders and $\mathbf{CC_3}$ contains only pulsers. Hence, the non-cyclic coded channel of Figure 28k is adequate. This confirms that the coded channel of Figure 37 is indeed adequate. Otherwise, the cyclic channel of Figure 30 would be needed. If this were so, the coded sequences of the main channel of Figure 37 would have to be recoded when they reached the end of the main channel and then transmitted to the beginning of the main channel, as is done in Figure 27.

The area **X** was designed next. The outputs o_2, o_3, and o_5 from the constructing unit **CU** pass through direct transmission channels in **Y**, enter pulsers in **X**, enter the main channel of **MC** as coded sequences, and pass through $\mathbf{CC_2}$ into **RWEC**. A completion signal destined for input i_3 of **CU** leaves **RWEC** at any one of the three exits shown in Figure 37, is coded in $\mathbf{CC_3}$, enters the main channel, is decoded in **X**, and passes through **Y** to the exit labeled "i_3". Area **X** thus contains three pulsers and one decoder, and is of height 36 and width 10. Von Neumann made area **Y** the same height as area **X**, and area **Z** the same width as area **X**, thus specifying the dimensions

of areas **Y**, **Z**, and **W**. The actual design of area **Y** is left until later, but it is very simple. Area **Y** contains four communication lines composed of ordinary transmission states; the rest of **Y** is unused.

We thus have the following dimensions:

Area **X**: 36 cells high, 10 cells wide
Area **Y**: 36 cells high, 31 cells wide
Area **Z**: 191 cells high, 10 cells wide
Area **W**: 191 cells high, 31 cells wide.

Von Neumann next considered the delays that are needed within **MC** to exclude the possibility of corruption by interference among the messages which circulate in this system. These fall into two classes: those needed to prevent corruption in the main channel and those needed to prevent corruption within **RWE** and **RWEC**. Von Neumann applied the rule (16′) of Section 3.6.1 and concluded that no corruption would occur. This conclusion is not correct, however.

Consider the pulses from outputs *e* and *f* of the control organ **CO** (Figs. 40 and 41). The pulse from *e* enters $\mathbf{CC}_3$ and thereby causes a coded sequence to enter the main channel; this coded sequence is decoded in $\mathbf{CC}_1$ and causes a periodic pulser of **RWE** to stop. The pulse from *f* usually enters $\mathbf{CC}_3$ and produces a coded sequence which eventually starts a pulser of **RWE**. The pulse from *f* also usually enters the top of the next control organ **CO**, exits from *c* (among other things), and enters $\mathbf{CC}_3$; this pulse eventually starts a triple-return counter of **RWE**. Hence the pulses coming from *e* and *f* of a **CO** normally cause three coded sequences to enter the main channel in close succession: the first sequence stops a periodic pulser of **RWE**, the second sequence starts a pulser of **RWE**, and the third sequence starts a triple-return counter of **RWE**. For some **CO** these sequences would overlap in the main channel if they were not properly delayed by means of delay paths between **RWEC** and $\mathbf{CC}_3$. There is ample room for these delay paths in **RWEC**.

A similar problem arises in **RWE**. Consider the input u_1 which feeds the connecting loop $\mathbf{C}_1$. The input u_1 is fed by both pulsers and periodic pulsers, and when loop $\mathbf{C}_1$ is being lengthened or shortened, a sequence of pulses from a pulser follows immediately after a sequence of pulses from a periodic pulser. These sequences must not overlap. Similar remarks apply to input v_1 of loop $\mathbf{C}_1$ and inputs u_2 and v_2 of loop $\mathbf{C}_2$ (see Sec. 4.2 and Table II of Sec. 4.3.2). This undesirable overlap of sequences can be prevented by adding delays in **RWE** or by changing the order of the periodic pulsers and pulsers in **RWE**.

These oversights are minor. A more important case of interference arises in the main channel when the periodic pulsers and triple-return counters of **RWE** are being used to lengthen or shorten the connecting loops $\mathbf{C}_1$ and $\mathbf{C}_2$. When the upper half of each loop is very roughly 200 cells long, a periodic pulser **PP** of **RWE** is started at about the same time as a triple-return counter (Φ_1 or Φ_2) emits an output pulse from its primary output b (Fig. 23). The coded sequence which starts the **PP** and the coded sequence coming from the decoder fed by Φ_1 or Φ_2 will overlap in the main channel, causing contamination. A full explanation of this interference problem, and its solution, will be given in Section 5.1.2.

The proper performance of the activities controlled by each of the 16 organs **CO** depends upon the observance of certain precisely specified delays, each one of these delays being specific to the particular **CO** involved. Each **CO** controls a lengthening or shortening operation on one of the connecting loops $\mathbf{C}_1$ or $\mathbf{C}_2$, and this is timed by means of the other connecting loop and a triple-return counter. We will illustrate this point by discussing the delays associated with control organ $\mathbf{CO}_3$; see Table V and Figures 39 and 41a.

The **PP**$(\overline{\overline{110110}})$ of **RWE** which has inputs $u_1 \cdot a_+$ and $u_1 \cdot a_-$ must be turned on for approximately $6n$ units of time, where x_n is the square of **L** under scan at the beginning of the process. The quantity $6n$ is unbounded and hence cannot be stored in the finite automaton **MC**. In Section 4.1.7 von Neumann introduced the timing loop $\mathbf{C}_2$ to store the quantity $2n$. A pulse takes approximately $6n$ time steps to travel around loop $\mathbf{C}_2$ three times; the delay is not precisely $6n$ because the loop $\mathbf{C}_2$ is lengthened or shortened before the loop $\mathbf{C}_1$ is lengthened or shortened. These three circuits around $\mathbf{C}_2$ are counted by the triple-return counter Φ_2. Clearly, the control interactions between $\mathbf{CO}_3$, on the one hand, and **PP**$(\overline{\overline{110110}})$ and Φ_2 on the other, take considerable time because **RWEC** is remote from **RWE**. Similar remarks apply to the other **CO**'s of **RWEC**.

We will enumerate the delays included in this control process, measuring them from the output of the confluent state in the upper right-hand corner of $\mathbf{CO}_3$.

(Δ_1) Starting from output c of $\mathbf{CO}_3$, the delay through $\mathbf{CC}_3$, the main channel, and $\mathbf{CC}_1$, ending at input $\Phi_2 \cdot a$ of **RWE**.

(Δ_2) The delay within Φ_2, the delay between Φ_2 and input v_2 of $\mathbf{C}_2$ (this is δ_2'); the delay between output w_2 of $\mathbf{C}_2$ and Φ_2 (this is δ_2''); and the delay within $\mathbf{C}_2$. Remember that a pulse goes around $\mathbf{C}_2$ three times. The total delay is $3(\delta_2' + \delta_2'') + a_2$ plus

the desired delay from the turning on to the turning off of **PP**($\overline{\overline{110110}}$); see Table III of Section 4.3.2.

(Δ_3) Starting from output $\Phi_2 \cdot b$ of **RWE**, the delay through $\mathbf{CC}_1$, the main channel, and $\mathbf{CC}_2$; the delay from d to e of $\mathbf{CO}_3$; and the delay through $\mathbf{CC}_3$, the main channel, and $\mathbf{CC}_1$, ending with input $u_1 \cdot a_-$ of **RWE**. This is approximately a complete loop around the memory control **MC**.

(Δ_4) The delay through area **D** (along delay path ℬ) of $\mathbf{CO}_3$; the delay from output b of $\mathbf{CO}_3$ through $\mathbf{CC}_3$ and the main channel to area **Z**; the delay through **Z**, **W**, and **Z** again; and the delay through the main channel and $\mathbf{CC}_1$ to input $u_1 \cdot a_+$ of **RWE**. The exact amount of the delay along ℬ and in **Z** and **W** has not been specified yet.

The memory control **MC** is 547 cells high and 87 cells wide, and so the delay around its main channel is considerable. It is clear that **PP**($\overline{\overline{110110}}$) must be started much later than Φ_2 because of the excess delay $3(\delta_2' + \delta_2'') + a_2$ plus the delay from the output of Φ_2 around the coded channel of **MC**, through **RWEC**, and back down the coded channel to the stop input $u_1 \cdot a_-$ of **PP**($\overline{\overline{110110}}$). Hence the delays in area **D** (path ℬ) of $\mathbf{CO}_3$ and the delays in areas **Z** and **W** are indeed necessary. Similar delays are needed for the other control organs **CO**, but the exact amount of delay needed varies with each **CO**, because the positions of the **CO**'s in **RWEC** and the positions of the **PP**'s and Φ's in **RWE** all vary.

Von Neumann made a lengthy and detailed calculation of the amount of delay needed in areas **D**, **Z**, and **W** for each **CO**. These calculations are exceedingly long and complicated because of the nature of von Neumann's design procedure. He had not yet specified the details of the design of **MC**: the size of area **D** of **CO** had not been fixed and the paths ℬ in the control organs $\mathbf{CO}_1$ through $\mathbf{CO}_{16}$ had not been specified; the exact code of the coded channel had not been chosen; and the exact locations of the organs of $\mathbf{CC}_2$, **RWEC**, **Z**, **W**, and **RWE** had not been specified. In particular, the height of **RWE** was determined by the height of $\mathbf{CC}_1$, and there is in fact much vacant space in **RWE**.

Von Neumann made his delay calculations with all these design parameters unspecified, and then specified these parameters on the basis of the results of his calculations. This is a very flexible design procedure, but it makes the calculations tedious and involved. Moreover, some of the dimensions von Neumann used at this point were wrong, notably the height of $\mathbf{CC}_1$; see the end of Section 4.3.3. These errors vitiate most of his delay calculations. For these reasons

we will not reproduce this part of the manuscript. Instead, we will summarize his chief results. These results show that von Neumann's design of the memory control **MC** does work, after certain modifications are made in it.

The delays involved depend on design details which are not yet fixed, and on design parameters which must be altered to correct for von Neumann's design errors. For these reasons we will give only very rough estimates of these delays. The delay $3(\delta_2' + \delta_2'') + a_2$ is very roughly 600 units. The delay Δ_3 from $\Phi_2 \cdot b$ of **RWE** through $\mathbf{CO}_3$ and back to $u_1 \cdot a_-$ of **RWE** is very roughly 1200 units. Note that according to von Neumann's calculation of the height of **RWE** (239 units—cf. Sec. 4.3.3), the upper $\mathbf{PP}(\overline{\overline{110110}})$ is about 200 units above Φ_2. The extra delay needed in areas **D**, **Z**, and **W** turns out to be very roughly 2000 units. The variation in delay from one **CO** to another is no more than about 100 units.

The common part of the extra delay associated with each periodic pulser of **RWE** (i.e., about 2000 units of delay) will be taken care of in areas **Z** and **W**. The part of the extra delay unique to each **CO** (i.e., about 100 units of delay) will be taken care of in the delay area **D** of that **CO**. At the end of Section 4.3.4 area **D** was allotted width 15, while its height h was left adjustable. The height of **RWEC** was controlled by the height of $\mathbf{CC}_3$, which is 545. A height of 545 for **RWEC** permits $h = 21$, so h is chosen to have this value. This gives **D** of **CO** an area of 21×15, or 315 cells, which is more than enough to accommodate that part of the extra delay which is unique to each **CO**.

Von Neumann now proceeds to design the areas **Z** and **W**. He divides **W** into four equal areas $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$, plus a slight excess. He divides **Z** into the corresponding parts $\mathbf{Z}_1$, $\mathbf{Z}_2$, $\mathbf{Z}_3$, and $\mathbf{Z}_4$. Areas $\mathbf{Z}_1$ and $\mathbf{W}_1$ are used to get the extra delay associated with the upper $\mathbf{PP}(\overline{\overline{110110}})$ of **RWE**, areas $\mathbf{Z}_2$ and $\mathbf{W}_2$ are used to get the extra delay needed for the upper $\mathbf{PP}(\overline{\overline{110000}})$ of **RWE**, and so forth for the other periodic pulsers of **RWE**.

Take $\mathbf{CO}_3$ as an example again. $\mathbf{CO}_3$ controls the upper $\mathbf{PP}(\overline{\overline{110110}})$ of **RWE**, starting it at input $u_1 \cdot a_+$ (Figs. 39 and 41a). In the design of Section 4.3.5, the start pulse leaves $\mathbf{CO}_3$ at $u_1 \cdot a_+$, is coded in $\mathbf{CC}_3$, travels through the main channel, is decoded at $\mathbf{CC}_1$, and enters **RWE** at input $u_1 \cdot a_+$. This design is to be modified now so as to add about 2000 units of delay to the path from $\mathbf{CO}_3$ to **RWE**; this extra delay is achieved by running the path through the areas $\mathbf{Z}_1$ and $\mathbf{W}_1$. To make this path pass through areas $\mathbf{Z}_1$ and $\mathbf{W}_1$, we code the output $u_1 \cdot a_+$ of $\mathbf{CO}_3$ into a new coded sequence $u_1^* \cdot a_+$, place a decoder for

$u_1^* \cdot a_+$ in $\mathbf{Z}_1$, place a pulser for $u_1 \cdot a_+$ in $\mathbf{Z}_1$, and connect the output of the decoder to the input of the pulser by a long path which travels through $\mathbf{W}_1$.

The delay path Δ_4 from $\mathbf{CO}_3$ to input $u_1 \cdot a_+$ of **RWE** is now as follows. A pulse leaves the confluent element at the upper right-hand corner of $\mathbf{CO}_3$ and travels along path $\mathfrak{B}$, exiting at $u_1^* \cdot a_+$, which is labeled $u_1 \cdot a_+$ in Figure 41a. A pulser of $\mathbf{CC}_3$ sends a coded sequence corresponding to $u_1^* \cdot a_+$ into the main channel; a decoder in $\mathbf{Z}_1$ detects this sequence and sends a pulse through a long path in area $\mathbf{W}_1$. This pulse stimulates a pulser in $\mathbf{Z}_1$ which sends a coded sequence corresponding to $u_1 \cdot a_+$ into the main channel. Finally, a decoder in $\mathbf{CC}_1$ detects this sequence and sends a pulse into the start input a_+ of the upper $\mathbf{PP}(\overline{\overline{110110}})$ of **RWE**.

Since control organs $\mathbf{CO}_5$, $\mathbf{CO}_7$, $\mathbf{CO}_{13}$, and $\mathbf{CO}_{15}$ also control the upper $\mathbf{PP}(\overline{\overline{110110}})$ of **RWE**, their outputs $u_1 \cdot a_+$ should also be changed to $u_1^* \cdot a_+$, and the pulsers which they stimulate in $\mathbf{CC}_3$ should be changed accordingly.

Similar arrangements should be made for the other **CO**'s of **RWEC** and the other parts of **Z** and **W**, so that a pulse from an output b of **CO** will pass through areas **Z** and **W** before starting a periodic pulser of **RWE**. For these arrangements coded sequences corresponding to $u_2^* \cdot a_+$, $u_3^* \cdot a_+$, and $u_4^* \cdot a_+$ are used.

In short, the following modifications are to be made in the design of **MC** as it was left at the end of Chapter 4. The labels $u_1 \cdot a_+$, $v_1 \cdot a_+$, $u_2 \cdot a_+$, and $v_2 \cdot a_+$ of Figure 41 are to be replaced by the labels $u_1^* \cdot a_+$, $v_1^* \cdot a_+$, $u_2^* \cdot a_+$, and $v_2^* \cdot a_+$, respectively, and the pulsers of $\mathbf{CC}_3$ recoded accordingly. Area $\mathbf{Z}_1$ has a decoder for $u_1^* \cdot a_+$ and a pulser for $u_1 \cdot a_+$, the output from the decoder feeding the pulser through a long delay path in area $\mathbf{W}_1$. Area $\mathbf{Z}_2$ has a decoder for $v_1^* \cdot a_+$ and a pulser for $v_1 \cdot a_+$, the former feeding the latter through area $\mathbf{W}_2$. Similarly, a decoder for $u_2^* \cdot a_+$ in $\mathbf{Z}_3$ feeds a pulser for $u_2 \cdot a_+$ via $\mathbf{W}_3$, and a decoder for $v_2^* \cdot a_+$ in $\mathbf{Z}_4$ feeds a pulser for $v_2 \cdot a_+$ via $\mathbf{W}_4$.

It will be recalled that von Neumann made an error in calculating the height of $\mathbf{CC}_1$; see the end of Section 4.3.3. Because of this error he thought that $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$ could each be of height 68. Their widths were to be 29, giving each an area of 1972 cells. He planned to attain the needed delay by running a path of ordinary transmission states back and forth through each area. He thus expected to obtain a delay of 1972 units of time, which is slightly more than he needed. Moreover, as he observed, additional delays could be obtained in the areas **D** of the **CO**'s. However, because of his error

in calculating the height of $\mathbf{CC}_1$, the areas $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$ can be of height 45 at most, which does not give enough delay.

There are a number of ways this error can be corrected. The area **W** can be extended to the right. The organs of **MC** can be rearranged to take advantage of the unused space in **X**, **Y**, **Z**, **RWE**, and $\mathbf{CC}_2$. But there are two more elegant ways of making the correction.

In the first place, each of the areas $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$ can be designed to give more than 1 unit of delay per cell. Since confluent states do not feed one another, they can be alternated with ordinary transmission states to give an average delay of $1\frac{1}{2}$ units per cell. An even larger delay per cell can be obtained by counting processes; we will indicate one way in which this can be done. The repeater of Figure 18e (Sec. 3.2.2) will by itself give a periodic sequence of the form $\overline{100\cdots00}$. Let the pulse to be delayed in area $\mathbf{W}_1$ start two of these repeaters, one producing a sequence of length 41 and the other a sequence of length 47, the numbers 41 and 47 being relatively prime. Feed the output from both repeaters to a confluent state. After 41×47 ($= 1927$) units of time there will be a coincidence at this confluent state. The resultant output pulse from the confluent state signals a delay of 1927 units, and this pulse can be used to turn off the two repeaters. In this way the delay of roughly 2000 units which von Neumann needs can be obtained in area $\mathbf{W}_1$, and similarly for areas $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$.

Another way of getting the needed delays in the area **W** that von Neumann actually had available is to replace the four delay paths of area **W** by a single path. To do this, break the connections from the main channel to the decoders of **Z**, and connect each decoder of **Z** directly to the pulser of **Z** which it formerly drove via a delay path in **W**. For example, the output from the decoder for $u_1^* \cdot a_+$ will now go directly into the pulser for $u_1 \cdot a_+$; see how $\mathbf{D}(\overline{11001})$ feeds $\mathbf{P}(\overline{10011})$ at the top of the coded channel of Figure 27. Then make a branch of the main channel which passes through the top of **Z**, goes through **W** to give a delay of about 2000 units in **W**, and then feeds the four decoders of **Z**.

The design of areas **Z** and **W** adds 4 new coded sequences to the coded channel, those symbolized by $u_1^* \cdot a_+$, $v_1^* \cdot a_+$, $u_2^* \cdot a_+$, and $v_2^* \cdot a_+$. There are 30 coded sequences associated with the decoders and pulsers of $\mathbf{CC}_1$ and 4 coded sequences associated with the decoders and pulsers of **X**. A coded sequence is associated with the output β of $\mathbf{CO}_{14}$; otherwise $\mathbf{CC}_2$ and $\mathbf{CC}_3$ do not add any sequences to the list already given. Thus the coded channel of the memory control **MC** needs 39 different coded sequences. At the end of Section 4.3.3

we tentatively assumed that a code with sequences of length 9 containing four ones, which allows 56 different sequences, would be adequate. This assumption is now confirmed.

This completes von Neumann's design of the memory control **MC**. The exact code of the coded channel has not been selected, and the exact positions of many of the organs of **MC** have not been chosen, but these are minor matters of detail.

Von Neumann concluded the manuscript by calculating the duration of the operations of the memory control **MC**. Equation (17′) of Section 4.1.3 specifies n^s as the number of the square x_n of the linear array **L** which is under scan at a given step s. Von Neumann found the total time for lengthening to be approximately $36n^s + 13{,}000$, and the total time for shortening to be approximately $48n^s + 20{,}000$.

5.1.2 Solution of the interference problem. The design of the memory control **MC** is now complete and workable except for the interference problem left unsolved in Section 5.1.1. In the present subsection we will explain this problem and give methods for solving it.

The problem involves the read-write-erase control **RWEC** and the read-write-erase unit **RWE**. In certain circumstances, a signal from **RWEC** to **RWE** interferes with a signal from **RWE** to **RWEC** in the main channel.

Von Neumann used n^s to represent the number of the square x_n of the linear array **L** which is under scan at a given step s. For certain values of n^s, a coded sequence used to start a periodic pulser of **RWE** will overlap a coded sequence coming from $\mathbf{CC}_1$ to instruct **RWEC** to stop that same periodic pulser. This overlap will occur in the main channel between $\mathbf{CC}_1$ and the end. In some cases this overlap will produce corruption by interference; see formula (16′) of Section 3.6.1. The two overlapping sequences will form a third sequence which will stimulate a decoder of $\mathbf{CC}_2$ that should not be stimulated at this point in the operation of **MC**.

This interference problem involves all of the periodic pulsers and triple-return counters of **RWE**, but for the sake of concreteness we will explain it in terms of the upper periodic pulser $\mathbf{PP}(\overline{\overline{110110}})$ of **RWE** and the triple-return counter Φ_2 used in timing it. Consider first the operation of $\mathbf{PP}(\overline{\overline{110110}})$. It is started with a pulse into its input $u_1 \cdot a_+$ at some time τ_+ and stopped with a pulse into its input $u_1 \cdot a_-$ at some time τ_-, where $\tau_+ < \tau_-$. Table V shows that $\tau_- - \tau_+$ equals approximately $6n^s$. Hence the time of the start at $u_1 \cdot a_+$ is given by

$$\tau_+ \doteq \tau_- - 6n^s. \tag{1}$$

Consider next the control of the stop at $u_1 \cdot a_-$. This is controlled by the pulse from the primary output $\Phi_2 \cdot b$ of the triple-return counter Φ_2 at some time τ_b. This pulse must travel from $\Phi_2 \cdot b$ through $\mathbf{CC}_1$, the main channel, $\mathbf{CC}_2$, some control organ of **RWEC** (e.g., $\mathbf{CO}_3$), the main channel again, and $\mathbf{CC}_1$ again, to stimulate $u_1 \cdot a_-$ and thereby stop $\mathbf{PP}(\overline{110110})$. The time required for this transit is the delay Δ_3 of Section 5.1.1, and is very roughly 1200 units. Hence the time of the exit from $\Phi_2 \cdot b$ of **RWE** is very roughly

$$\tau_b \doteq \tau_- - 1200. \tag{2}$$

Now compare equations (1) and (2), keeping in mind that n^s is the number of the square of **L** under scan. For $n^s = 0$, τ_+ is about 1200 units larger (later) than τ_b. For large n^s, τ_+ is very much smaller (earlier) than τ_b. And for n^s equal to about 200, the times τ_+ and τ_b are equal. Hence when n^s equals approximately 200, the coded sequence $u_1 \cdot a_+$ will be in the main channel near $\mathbf{CC}_1$ at about the same time the coded sequence $\Phi_2 \cdot b$ enters this part of the main channel. The result will be corruption by interference in the coded channel.

Hence von Neumann's design of **MC** does not work correctly when the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are of certain lengths. We will give two different solutions to this interference problem. The first consists in avoiding the interference by never using those cells of **L** where it arises. This may be done by programming **MC** in such a way that no value of n^s less than, say, 250 is used. The cells $x_0, x_1, \cdots, x_{249}$ of **L** will then never be used. This implies that when **MC** begins life, loop $\mathbf{C}_1$ will pass through cell x_{250} and loop $\mathbf{C}_2$ has a corresponding length. And this in turn affects the arrangements for universal construction, as we shall now see.

Von Neumann's arrangements for universal construction are as follows (Secs. 1.5.2 and 1.7.2.1). Enclose the secondary automaton to be constructed in a rectangular area of side lengths α and β. The desired state of each cell (i, j) $(i = 0, 1, \cdots, \alpha - 1; j = 0, 1, \cdots, \beta - 1)$ of this rectangular area is represented by λ_{ij}. The construction process will leave an arbitrary cell (i, j) in any 1 of the 10 unexcited states **U**, $\mathbf{T}_{u\alpha 0}$ $(u = 0, 1; \alpha = 0, 1, 2, 3)$, and $\mathbf{C}_{00}$ (see Figs. 9 and 10), and so λ_{ij} is limited to 10 values. The desired construction is described to the primary (parent) automaton by giving it the sequence $\lambda_{00}, \lambda_{01}, \cdots, \lambda_{0(\beta-1)}, \lambda_{10}, \cdots, \lambda_{ij}, \cdots, \lambda_{(\alpha-1)(\beta-1)}$.

The secondary automaton to be constructed will in general have an indefinitely extendible linear array **L** and a memory control **MC** to control it. This being so, von Neumann designed **MC** so that it would be rectangular. Now on this first solution of the interference

problem, **MC** must be constructed so that initially loop $\mathbf{C}_1$ will pass through cell x_{250} and loop $\mathbf{C}_2$ has a corresponding length. To keep the rectangular shape of **MC**, we extend the boundary of **MC** 250 cells to the right, making **MC** 337 cells wide and 547 cells high (see Fig. 37). All but about 1 per cent of the cells in the added area will always be in the unexcitable state **U**, which means that this new area is being used very inefficiently.

Hence while this first solution to the interference problem is workable, it is most inelegant, and von Neumann would not have been satisfied with it. For this reason we will suggest a second solution. This solution is also of general interest because it illustrates some fundamental features of von Neumann's cellular structure.

Von Neumann's cellular structure consists of an infinite iterated array of the same 29-state finite automaton. Any finite number of these automata can be active at a given time; therefore the cellular structure is capable of an indefinitely large amount of concurrent action or parallel data processing. In his design of the self-reproducing automaton, von Neumann was not taking advantage of the parallel data processing capability of his cellular structure. Rather, he was designing the self-reproducing automaton so that for the most part it would do only one thing at a time. In this respect, von Neumann's logical design of the self-reproducing automaton is similar to his logical design of the EDVAC (see pp. 10–11 above). Moreover, when he did use parallel activity in the lengthening and shortening of loops $\mathbf{C}_1$ and $\mathbf{C}_2$, he ran into timing problems. He needed the delay area **D** of each **CO** (Fig. 40) and the delay area **W** of **MC** (Fig. 37) to postpone the start of a periodic pulser of **RWE**, and in arranging for the stop of this periodic pulser he ran into the interference problem we are now discussing.

This interference problem would not arise if two signal paths (wires) could be crossed without intersecting; so let us look at the possibilities for wire-crossing in von Neumann's cellular structure and variations of it. Wires could cross naturally without touching in a 3-dimensional cellular structure, but von Neumann wanted to construct a self-reproducing automaton in 2 dimensions (Sec. 1.3.3.3). Keeping to his 2-dimensional structure, he could have incorporated a crossing primitive into each cell. For example, his 29-state automaton could be augmented by a new primitive consisting of $\mathbf{T}_{00\epsilon}$ and $\mathbf{T}_{01\epsilon'}$ together, so that information could flow through a cell from left to right and from bottom to top simultaneously.[1] Von Neumann did not

[1] Cellular structures which have crossing primitives are considered in Church, "Application of Recursive Arithmetic to the Problem of Circuit

say why he didn't include a crossing primitive among the states of his basic automaton, but it was probably because he wished to keep the number of states small.

It is actually possible to synthesize a crossing organ in von Neumann's cellular structure. Such an organ is shown in Figure 42a.[2] The symbolism of this and subsequent figures is somewhat different from von Neumann's symbolism. A single arrow is used to represent an ordinary transmission state; later a double arrow will be used to represent a special transmission state. A dot besides an arrow indicates that the cell is initially active, i.e., is excited at time $t = 0$. Figure 42b gives the initial state of each of the five "clocks" of the crossing organ in von Neumann's notation.

Consider first the behavior of the crossing organ when the inputs a_1 and a_2 are 000 $\cdots$. The clocks send alternate zeros and ones into both inputs of each of the six confluent states *C3*, *C6*, *F3*, *F6*, *E1*, and *H5*. The phasing of the ones (i.e., pulses) as they enter these confluent cells is indicated in Figure 42a by dashed and dotted lines. A dashed line signifies a one (pulse) at every even time ($t \geqq 4$) and a dotted line signifies a one (pulse) at every odd time ($t \geqq 3$). It is clear from Figure 42a that the two sequences arriving at a confluent state are out of phase. The function of these clock sequences is to gate the sequences coming into a_1 and a_2 so that they can cross each other.

The sequence i_0 , i_1 , i_2 , i_3 , i_4 , i_5 , $\cdots$ entering a_1 is split into two sequences by the confluent cell *A4*. The clocks insert ones into every even position of the upper sequence and into every odd position of the lower sequence. The odd bits —, i_1 , —, i_3 , —, i_5 , $\cdots$ are allowed by the gating pulses to pass along row 3 and out at b_1 , while the even bits i_0 , —, i_2 , —, i_4 , —, $\cdots$ are allowed by the gating pulses to pass along row 6 and out at b_1 . Similarly, the sequence j_0 , j_1 , j_2 , j_3 , j_4 , j_5 , $\cdots$ entering a_2 is split, with the even bits j_0 , —, j_2 , —, j_4 , —, $\cdots$ traveling up column *C* and the odd bits —, j_1 , —, j_3 , —, j_5 , $\cdots$ traveling up column *F*. The phasing of the whole system is such that the sequence j_0 , —, j_2 , —, j_4 , —, $\cdots$ is interleaved with i_0 , —, i_2 , —, i_4 , —, $\cdots$ at cell *C6* and with —, i_1 , —, i_3 , —, i_5 , $\cdots$ at cell *C3*. Likewise, the sequence —, j_1 , —, j_3 , —, j_5 , $\cdots$ is inter-

---

Synthesis." Cellular structures which have crossing primitives and also permit the construction of zero-delay paths of unbounded (but finite) length are discussed in Burks, "Computation, Behavior, and Structure in Fixed and Growing Automata," Holland, "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously," and Holland, "Iterative Circuit Computers."

[2] The crossing organ of Figure 42 was designed by J. E. Gorman.

leaved with i_0 , —, i_2 , —, i_4 , —, $\cdots$ at cell *F6* and with —, i_1 , —, i_3 , —, i_5 , $\cdots$ at cell *F3*. For example, the sequences entering and leaving cell *C6* are:

From the left:	$0\ 0\ 1\ 0\ 1\ i_0\ 1\ i_2\ 1\ i_4 \cdots$
From below:	$0\ 0\ 0\ 1\ j_0\ 1\ j_2\ 1\ j_4\ 1 \cdots$
Output:	$0\ 0\ 0\ 0\ 0\ 0\ j_0\ i_0\ j_2\ i_2\ j_4\ i_4 \cdots$

The sequences from cells *C3* and *F3* are combined in cell *E1* to give the output j_0 , j_1 , j_2 , $\cdots$ delayed 15 units of time. Similarly, the sequences from cells *F3* and *F6* are combined in cell *H5* to give the output i_0 , i_1 , i_2 , $\cdots$ delayed 15 units of time. In this way information passes from a_1 to b_1 and from a_2 to b_2 without any cross interference.

We shall now employ the crossing organ to solve the interference problem in von Neumann's design of the memory control **MC**. The interference occurs when n^s is very roughly 200. Under this condition, a coded sequence $u_1 \cdot a_+$, $v_1 \cdot a_+$, $u_2 \cdot a_+$, or $v_2 \cdot a_+$ which is to start a periodic pulser of **RWE** interferes with a coded sequence $\Phi_2 \cdot b$ or $\Phi_1 \cdot b$ which should eventually (via **RWEC**) stop this same periodic pulser. By means of the crossing organ the signal $\Phi_2 \cdot b$ can be sent directly from Φ_2 to stop an upper periodic pulser of **RWE** and can then be sent into the coded channel to signal **RWEC** that this phase of the lengthening or shortening of loop $\mathbf{C}_1$ is finished. Likewise, the signal $\Phi_1 \cdot b$ can be sent directly from Φ_1 to stop a lower periodic pulser of **RWE** and thence into the coded channel. This modification of von Neumann's design greatly reduces the amount of delay needed in the delay areas **D** (of a **CO**) and **W** (of **MC**).

The simplest way to arrange for a direct stop of a periodic pulser by a triple-return counter is this.

(1) In Figure 39a, exchange the pulser labeled "0.0" with **PP**$(\overline{\overline{110000}})$, and invert **PP**$(\overline{\overline{110110}})$. The stop inputs of both periodic pulsers can now be connected; call the point of connection $u_1 \cdot v_1 \cdot a_-$. Similarly, in Figure 39b exchange the pulser labeled "0.4" with **PP**$(\overline{\overline{110000}})$ and invert **PP**$(\overline{\overline{110110}})$; call the common stop input of these two periodic pulsers $u_2 \cdot v_2 \cdot a_-$.

(2) Replace the single column of **U** states located between the upward and downward portions of the main channel near the center of **MC** by 12 columns of cells. This room will be used for the channels and crossing organs of (3) and (4) below.

(3) Then take the primary output b of Φ_1 across the main channel (by means of a crossing organ), down, back across the main channel (by means of a second crossing organ), into $u_2 \cdot v_2 \cdot a_-$, and also into

a pulser of $\mathbf{CC}_1$ (which will signal the appropriate **CO** of **RWEC** that this phase of the lengthening or shortening of loop $\mathbf{C}_2$ is finished).

(4) Finally, take the primary output b of Φ_2 across the main channel (by means of a third crossing organ), up and outside of the channel of (3), back across the main channel (by means of a fourth crossing organ), into $u_1 \cdot v_1 \cdot a_-$, and also into a pulser of $\mathbf{CC}_1$ (which will signal the appropriate **CO** of **RWEC** that this phase of the lengthening or shortening of loop $\mathbf{C}_1$ is finished).

These arrangements take only four crossing organs and considerably reduce the delay circuitry of von Neumann's design. But they raise a problem concerning the construction of the memory control **MC**, and hence of the construction of any secondary automaton which contains an indefinitely extendible linear array **L**. The construction process von Neumann envisaged has two stages (cf. Secs. 1.5.2 and 1.7.2.1). First, the primary (constructing) automaton puts each cell of the secondary area in 1 of the 10 unexcited (quiescent) states **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), and $\mathbf{C}_{00}$ (see Figs. 9 and 10). Von Neumann called the result of this process the *initial state* of the secondary automaton. Second, the primary automaton stimulates the secondary automaton at an appropriate point on its periphery, so that the secondary may begin its intended activity. Von Neumann called this starting signal the secondary automaton's *starting stimulus*.

Thus the secondary automata von Neumann wished to have constructed by the primary constructing automaton (i.e., from within his cellular structure) are all of a special kind: each cell of the secondary is *initially* in 1 of the 10 quiescent states **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), and $\mathbf{C}_{00}$; and the automaton may be started by a stimulus on its periphery. With this in mind, we define an *initially quiescent automaton* to be a finite area of von Neumann's 29-state cellular structure every cell of which is in 1 of the 10 quiescent states **U**, $\mathbf{T}_{u\alpha 0}$, and $\mathbf{C}_{00}$.

Since the crossing organ contains excited ordinary transmission states and excited confluent states, any automaton containing it is not initially quiescent. Consequently, the design of the memory control **MC** must be further modified so that **MC** will contain no excited cells initially, but that before it is used **MC** will contain the four crossing organs described above.

The following procedure will work. Further enlarge the area in the center of **MC** to allow for the four constructing devices and four decoders described below. Modify each of the four crossing organs of **MC** as follows: replace each active state ($\mathbf{T}_{011}$, $\mathbf{C}_{10}$, $\mathbf{C}_{01}$) of Figure 42 by the corresponding passive state ($\mathbf{T}_{010}$, $\mathbf{C}_{00}$, $\mathbf{C}_{00}$), and delete cells

A5, *B5*, *C5* (replace them by **U**'s) to make the center clock accessible from the outside. Then provide, for each modified crossing organ, a special-purpose constructing device and a decoder to start it. The constructing device will employ the general procedure for modifying a remote cell which was described on p. 155 and illustrated in Figure 14; by means of this procedure it can start the five clocks of the crossing organ in phase and rebuild cells *A5*, *B5*, and *C5*. The constructing device itself will be started by the decoder on signal from the main channel.

Now arrange for the secondary automaton's starting stimulus to put a (new) coded sequence into the main channel of **MC**. This sequence will be sensed by each of the four decoders, which will in turn start the four constructing devices. Each constructing device will send a constructing arm to its crossing organ, start each of the five clocks of the crossing organ in proper phase, rebuild cells *A5*, *B5*, *C5* of Figure 42, and leave the neighborhood of the crossing organ in the proper state. The memory control **MC** is then ready to operate, and the secondary automaton containing it can proceed in its normal way.

This completes the second solution of the interference problem in von Neumann's design of the memory control **MC**. Neither solution is ideal: the first solution is inelegant, and the second solution is rather complicated. A radically different and much better approach to the design of **MC** will be suggested in Section 5.2.2. However, the inelegance or complexity of the final design of **MC** is not directly relevant to von Neumann's central purpose. Von Neumann was seeking an existence proof of self-reproduction, that is, a proof that self-reproduction is possible in his cellular structure (Sec. 1.1.2.1). The construction of the memory control **MC** is a step towards this proof, and for this purpose it suffices that there exists a workable **MC**.

Let us summarize the results which have been achieved so far. The unlimited linear array **L** together with its control **MC** is a tape unit with an unlimited memory capacity. It may be constructed (embedded) in the 29-state cellular structure as an initially quiescent automaton. Hence, *an initially quiescent automaton which performs the functions of a tape unit with unlimited memory capacity can be embedded in von Neumann's 29-state cellular structure.*

5.1.3 Logical universality of the cellular structure. Let us next review briefly how von Neumann planned to use a tape unit with unlimited memory capacity in his cellular structure.

He discussed Turing's universal automaton in the Second Lecture of Part I of the present work. Early in Part II he stated the five main

questions considered in this part: (A) *Logical universality:* can any single automaton perform all the logical operations which are performable with finite (but arbitrarily extensive) means? (B) *Constructibility:* can automata be constructed by other automata? (C) *Construction universality:* can any single automaton construct every other automaton? (D) *Self-reproduction:* can any automaton construct copies of itself? (E) *Evolution:* can the construction of automata by automata progress from simple automata to increasingly complicated automata? Von Neumann stated that Turing had answered the first question. That is, Turing's universal computing automaton (machine) is logically universal. Von Neumann then promised to establish affirmative answers to questions (B)–(D).

In discussing general construction schemes, von Neumann introduced the unlimited memory array **L** and its "ancillary observation, exploration, and construction facilities" (Sec. 1.4.2.3). The latter is the memory control **MC**, the design of which we have just completed (Sec. 5.1.2). Let us call the complex **L** + **MC** a "tape unit." In Section 1.4.2.2 von Neumann stated in effect that this tape unit can be used as the unlimited memory of a constructing automaton, in Section 1.5 he indicated how to use it as the unlimited memory of a universal constructing automaton, and in Section 1.6 he indicated how to use a universal constructing automaton to obtain self-reproduction. We will show in Sections 5.2 and 5.3 how these results may be achieved.

In Section 1.4.2.3 von Neumann stated that the tape unit can be used as unlimited memory of a logically universal automaton or universal Turing machine. Earlier, in the Second Lecture of Part I above, he explained how a universal Turing machine works. And in Sections 4.1.3 and 4.1.4 he outlined how the constructing unit **CU** can operate the tape unit **MC** + **L**. Putting all these ideas together, we will show how to design a universal Turing machine in von Neumann's cellular structure.

A Turing machine has two main parts: a tape unit with an indefinite memory capacity, and a finite automaton which can interact with this tape unit. As outlined in Sections 4.1.3 and 4.1.4 above, a constructing automaton has two corresponding parts: a tape unit **MC** + **L** and a constructing unit **CU** which directs the construction of a secondary automaton on the basis of the information stored in **L**. Thus **CU** is a finite automaton which interacts with the tape unit and also performs the function of construction. Hence our task is to adapt von Neumann's outline for the interaction of **CU** and **MC** + **L** to cover the operation of a Turing machine.

A finite automaton **FA** has a finite number of states $\alpha = 1, \cdots, a$. The automaton **FA** and the tape unit **MC** + **L** operate in a succession of steps 0, 1, 2, 3, $\cdots$, s, $s + 1$, $\cdots$. Let the state of **FA** at step number zero be state number one, let x_{n^0} be the cell of **L** under scan at the beginning of step s, and let $\xi^s_{n^s}$ $(= 0, 1)$ represent the content of x_{n^s} at this time. The finite automaton **FA** is then defined by three functions A, X, and E:

The function A specifies the next state α^{s+1} as a function of present state α^s and the contents $\xi^s_{n^s}$ of x_{n^s} at the beginning of step s: $\alpha^{s+1} = A(\alpha^s, \xi^s_{n^s})$.

The function X specifies the number $\xi^{s+1}_{n^s}$ to be written in x_{n^s} as a function of α^s and $\xi^s_{n^s}$, or equivalently, as a function of α^{s+1}: $\xi^{s+1}_{n^s} = X(\alpha^{s+1})$.

The function E specifies the value of the lengthening-shortening parameter ϵ^{s+1} as a function of α^{s+1}: $\epsilon^{s+1} = E(\alpha^{s+1})$.

The ranges of arguments and function values are:

α^s and α^{s+1} range over the finite automaton states 1, 2, $\cdots$, a;

$\xi^s_{n^s}$ and $\xi^{s+1}_{n^s}$ range over 0 (representing "0" on x_n) and 1 (representing "1" on x_n);

ϵ^{s+1} ranges over $+1$ (lengthening) and -1 (shortening).

An "initially quiescent automaton" is one in which every cell is initially in 1 of the 10 quiescent states **U**, $\mathbf{T}_{u\alpha 0}$, and $\mathbf{C}_{00}$ (Sec. 5.1.2). We will show next how to embed any given finite automaton **FA** in von Neumann's cellular structure, that is, how to construct an initially quiescent automaton which will simulate **FA**.

Each of the a states of **FA** will be represented by a copy of the state organ **SO** of Figure 43. The periodic pulsers **PP** $(\bar{\bar{1}})$ of this figure are the alternate periodic pulsers of Figure 20, which can be turned off without harm even when they have not been turned on. These **PP** $(\bar{\bar{1}})$ are not shown to scale in Figure 43; each is actually 13 cells long and 4 cells high. As in Figure 42, single arrows are used to represent ordinary transmission states.

The finite automaton **FA** consists of a copies of **SO** interconnected by means of a coded channel (Sec. 3.6 and Fig. 27). The specific interconnections within an **FA** are determined by its three functions A, X, and E. Within **FA**, control is shifted from one state organ $\mathbf{SO}_\alpha$ to another state organ $\mathbf{SO}_{\alpha'}$ in accordance with the information **FA** receives from **MC**. The automata **FA** and **MC** are interconnected by eight channels. The inputs i_1, i_2, i_3 of **FA** come from the outputs of **MC** which are so labeled (Fig. 37); the outputs o_1, o_2, o_3, o_4, and o_5 of **FA** go to the inputs of **MC** which are so labeled.

The relation of the **SO**'s of **FA** to the coded channel of **FA** is very

much the same as the relation of the **CO**'s of **MC** to the coded channel of **MC** (Fig. 37).

We will explain how the composite **FA** + (**MC** + **L**) operates. At the beginning of step s the following conditions obtain:

(1a) A stimulus is arriving at input i_3 of **FA**, signifying that step $s - 1$ has been completed and it is time to begin step s.

(1b) Cell x_{n^s} of **L** is under scan; i.e., **MC** is connected to x_{n^s} via the connecting loop $\mathbf{C}_1$. The contents of the cell x_{n^s} are designated to be $\xi^s_{n^s}$ $(= 0, 1)$.

(1c) **FA** is in state α^s; i.e., the state organ $\mathbf{SO}_{\alpha^s}$ of **FA** is in control. More specifically, the upper $\mathbf{PP}(\bar{1})$ of Figure 43 is active. This periodic pulser has been on (active) while loops $\mathbf{C}_1$ and $\mathbf{C}_2$ of Figure 37 were being shortened or lengthened at the end of step $s - 1$.

The reading process is now inaugurated and control is shifted to the lower $\mathbf{PP}(\bar{\bar{1}})$ of $\mathbf{SO}_{\alpha^s}$:

(2a) The pulse entering input i_3 of **FA** goes via the coded channel to the input b of each **SO** of **FA**. It affects only $\mathbf{SO}_{\alpha^s}$, turning off its upper $\mathbf{PP}(\bar{\bar{1}})$, and passing through the confluent state turned on by this $\mathbf{PP}(\bar{\bar{1}})$ to exit from outputs j and k.

(2b) The stimulus from exit j goes via the coded channel of **FA** to output o_1 of **FA**, and thence to input o_1 of **MC**, where it starts the process of reading cell x_{n^s} of **L**.

(2c) The stimulus from exit k of $\mathbf{SO}_{\alpha^s}$ goes via the coded channel of **FA** to input f of this same $\mathbf{SO}_{\alpha^s}$, where it starts the lower $\mathbf{PP}(\bar{\bar{1}})$. This $\mathbf{PP}(\bar{\bar{1}})$ will be active while cell x_{n^s} is being read by **MC**.

The tape unit **MC** + **L** then reads the contents $\xi^s_{n^s}$ of cell x_{n^s} and sends the result to **FA**:

If $\xi^s_{n^s} = 0$ (i.e., x_{n^s} stores a "zero"), then a stimulus goes from output i_1 of **MC** to input i_1 of **FA**.

If $\xi^s_{n^s} = 1$ (i.e., x_{n^s} stores a "one"), then a stimulus goes from output i_2 of **MC** to input i_2 of **FA**.

We state next how these signals affect **FA**, using brackets to indicate the effects of $\xi^s_{n^s} = 0$ and braces to indicate the effects of $\xi^s_{n^s} = 1$.

(3a) The [input i_1] {input i_2} goes via the coded channel of **FA** to enter [input d] {inputs e and g} of every **SO**. These signals affect only $\mathbf{SO}_{\alpha^s}$.

(3b) The pulse from [i_1] {i_2} turns off the lower $\mathbf{PP}(\bar{\bar{1}})$ of $\mathbf{SO}_{\alpha^s}$.

(3c) The pulse from [i_1] {i_2} into [d] {g} is passed by the [upper] {lower} confluent state which is turned on by the output of the lower $\mathbf{PP}(\bar{\bar{1}})$, and exits from outputs [l, m, n] {p, q, r}.

The pulses from $[l, m, n]$ $\{p, q, r\}$ will determine the next state of **FA**, what is to be written in x_{n^s}, and whether loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be lengthened or shortened.

The specific connections of these outputs l, m, n, p, q, and r are determined by the three functions A, X, and E which characterize the given finite automaton **FA**. For each state organ $\mathbf{SO}_{\alpha^s}$ ($\alpha^s = 1, 2, \cdots, a$), these connections are made as follows:

(4a) The pulse from $[l]$ $\{p\}$ goes via the coded channel of **FA** to input c of $\mathbf{SO}_{\alpha^{s+1}}$, where $\alpha^{s+1} = A(\alpha^s, \xi_{n^s}^s)$.

(4b) The pulse from $[m]$ $\{q\}$ goes via the coded channel and outputs of **FA** to:
input o_2 of **MC** if $\xi_n^{s+1} = 0$ (i.e., "0" is to be written in x_{n^s}),
input o_3 of **MC** if $\xi_n^{s+1} = 1$ (i.e., "1" is to be written in x_{n^s}),
where $\xi_n^{s+1} = X(\alpha^{s+1})$.

(4c) The pulse from $[n]$ $\{r\}$ goes via the coded channel and outputs of **FA** to:
input o_4 of **MC** if $\epsilon^{s+1} = 1$ (i.e., loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be lengthened),
input o_5 of **MC** if $\epsilon^{s+1} = -1$ (i.e., loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be shortened),
where $\epsilon_{n^s}^{s+1} = E(\alpha^{s+1})$.

This completes our construction of an arbitrary finite automaton **FA** in von Neumann's cellular structure, except for specifying the state for step $s = 0$. This state was stipulated to be state number one and hence is represented by the state organ $\mathbf{SO}_1$. If the confluent cell in the upper right-hand corner of $\mathbf{SO}_1$ were initially in the state $\mathbf{C}_{10}$, starting pulses would emerge from exits j and k of $\mathbf{SO}_1$. These pulses would stimulate pulsers of the coded channel which in turn would inject coded sequences γ_j and γ_k into the main channel, and **FA** would then operate as described above. However, such a device would not be an initially quiescent automaton, for its initial state would contain one cell in a state other than **U**, $\mathbf{T}_{u\alpha 0}$, and $\mathbf{C}_{00}$. To make **FA** initially quiescent, we merely arrange for its starting stimulus to stimulate pulsers of the coded channel which will also inject the coded sequences γ_j and γ_k into the main channel of **FA**.

This concludes our demonstration that an arbitrary finite automaton can be embedded in von Neumann's cellular structure as an initially quiescent automaton. We saw at the end of Section 5.1.2 that a tape unit with unlimited memory capacity can be embedded in this cellular structure as an initially quiescent automaton. Hence any arbitrary Turing machine can be embedded as an initially

quiescent automaton, and a fortiori a universal Turing machine can be embedded as an initially quiescent automaton.

It should be noted that all these embedded devices will operate slowly with respect to the temporal reference frame of the cellular structure. This temporal reference frame consists of the discrete time steps $t = 0, 1, 2, \cdots$, and the fundamental operations of the 29-state automata of the cells take place in this reference frame (Secs. 1.2.1 and 1.3.3.5). The finite automata and Turing machines which are embedded in the cellular structure operate in this temporal reference frame too, but the succession of steps $s = 0, 1, 2, \cdots$ takes place more slowly. In general, each of the steps s takes several time units t. In the case of a tape unit **MC** + **L**, the steps s take longer and longer as the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ get longer and longer.

Finite automata and Turing machines are usually regarded as devices which accomplish one step s in one unit of time t. In other words, $s = t$, and the machines perform their computations in "real time." The initially quiescent automata of von Neumann's cellular structure which simulate finite automata and Turing machines do not operate at the same rate as the devices they simulate, but they compute the same results.[3]

Von Neumann's main aim was to achieve construction, construction universality, and self-reproduction in his 29-state infinite cellular structure (Sec. 1.1.2.1). Let us summarize what has been accomplished so far, and how it bears on his main aim.

We have shown how to embed in von Neumann's 29-state cellular structure an initially quiescent automaton which performs the computations of a universal Turing machine. Hence this cellular structure is logically universal.

In planning his universal constructing automaton, von Neumann was guided by the parallel notion of a universal computing machine. His universal constructing automaton will operate like a universal computing machine, the chief difference being that the output of the computing machine is a computation, while the output of the constructing automaton is a sequence of signals which constructs an initially quiescent secondary automaton. A universal computing machine M_u is a complex **FA** + (**MC** + **L**) with this property:

[3] The real time performance of an automaton is usually called its "behavior" and is to be contrasted with the computed answer or "computation" of an automaton. For a discussion of behavior and computation in finite and infinite systems see Burks, "Toward a Theory of Automata Based on More Realistic Primitive Elements," McNaughton, "On Nets Made up of Badly Timed Elements," and Holland, "Universal Embedding Spaces for Automata."

for each Turing machine M, there is a coded description $\mathfrak{D}'(M)$ such that, when $\mathfrak{D}'(M)$ is stored on **L**, M_u will simulate M, that is, M_u will compute the same result that M computes. Analogously, the universal constructor M_c is a complex **CU** + (**MC** + **L**) with this property: for each initially quiescent secondary automaton M, there is a coded description $\mathfrak{D}(M)$ such that, when $\mathfrak{D}(M)$ is stored on **L**, M_c will construct M.[4]

Thus the essential step remaining in von Neumann's program to achieve construction, construction universality, and self-reproduction in his cellular structure is to design the construction unit **CU**.

5.2 The Universal Constructor CU + (MC + L)

5.2.1 The constructing arm. Von Neumann's universal constructing automaton consists of a constructing unit **CU** combined with a tape unit **MC** + **L** (Sec. 4.1.1). He calls this the "primary automaton," and he calls the initially quiescent automaton to be constructed the "secondary automaton" (Sec. 1.4). Since the tape unit has been designed (Sec. 5.1.2), it remains to design the constructing unit.

We will first discuss the overall arrangements for construction. See Figure 50. This figure is not drawn to scale, the universal constructor being very much larger than shown here (cf. Fig. 37). Also, the loop for reading **L** is not shown, nor the arrangement for lengthening and shortening the loop (cf. Figs. 37 and 51).

Some point on the universal constructor is designated as the origin (0,0) of a coordinate system for locating the secondary automaton. The secondary automaton occupies a rectangular area of width α and height β, the lower left-hand corner of which is located at x_1, y_1 with respect to the origin (Fig. 50).[5] For the sake of simplicity we will confine the secondary automaton to the first quadrant, so that $x_1 \geqq 0$ and $y_1 \geqq 0$. The internal structure of the secondary automaton is completely characterized by the sequence λ_{ij} for $i = 0, \cdots, \alpha - 1$ and $j = 0, \cdots, \beta - 1$, where each λ_{ij} specifies one of the quiescent states **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), and $\mathbf{C}_{00}$.

[4] Compare the discussion of a universal Turing machine in the Second Lecture of Part I with the discussion of the universal constructor in Sec. 5.3.1 below. See also Burks, "Programming and the Theory of Automata."

The description of $\mathfrak{D}'(M)$ which is given to a universal Turing machine is usually coded differently than the description $\mathfrak{D}(M)$ which is given to a universal constructor. $\mathfrak{D}'(M)$ is coded in terms of the states of the finite part of M, whereas $\mathfrak{D}(M)$ is coded in terms of the initial states of the cells of M. See the next section.

[5] Cf. Sec. 1.5.2. In von Neumann's reference system, a point such as (x_1, y_1) is located at the center of a cell rather than at a corner of a cell.

The general procedure for construction is this. A coded representation of x_1, y_1, α, β, and the λ_{ij} is placed on the linear array **L**. The constructing unit **CU** reads this information with the aid of the memory control **MC**, interprets the information, and acts on it. The primary automaton operates on the (possibly remote) secondary area by means of a "constructing arm" or information path which extends from the primary automaton to the secondary area. The constructing unit **CU** first builds the constructing arm out to the secondary area. Then **CU** sends signals down the arm which construct the secondary automaton and provide its starting stimulus. Finally, **CU** withdraws the constructing arm.

It is clear that the first step in designing the constructing unit **CU** is to design the constructing arm which will be operated by **CU** under control of the information received from the tape unit **MC** + **L**. Actually, we have essentially used such a constructing arm before. Figure 14 illustrates a procedure for extending a constructing path, modifying a remote cell, and withdrawing the constructing path. This same procedure is used for lengthening and shortening the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ and writing in cell x_n of the linear array **L** (Sec. 4.2 and Figs. 32–36). In this case, the upper half of each loop constitutes the constructing path.

In both of these cases a single path of transmission states (sometimes ordinary, sometimes special) goes from the constructing device to the area of construction. For this reason we will call this von Neumann's "single path construction procedure." Let us look at this procedure in detail to see if it can be employed for the constructing arm needed now.

The construction of the secondary automaton takes place at the end of the constructing path, in accordance with the rules summarized in Figure 9. The two processes involved are the direct process (for construction) and the reverse process (for destruction). Destruction is as necessary as construction, for the operating terminus of the construction path can be withdrawn only by changing a transmission state $\mathbf{T}_{u\alpha 0}$ into an unexcitable state **U**. Construction can be accomplished in a given cell by feeding pulses into this cell from either an ordinary or a special transmission state, but destruction requires the proper kind of transmission state: special kills ordinary but not special, and ordinary kills special but not ordinary. This distinction is necessary if information signals are to be distinguished from destruction signals (Sec. 2.6.2.2).

Thus the operations at the end of the constructing path sometimes require special transmission states and sometimes require ordinary

transmission states. In Von Neumann's single path construction procedure this is accomplished as follows. The beginning of the construction path is fed by both an ordinary transmission state (e.g., cell *B1* of Fig. 14) and by a special transmission state (e.g., cell *B3* of Fig. 14). Whenever the cell at the operating end (head) of the constructing path needs to be changed from an ordinary transmission state to a special transmission state (or vice versa), the whole path is changed from ordinary transmission states to special transmission states (or vice versa). Compare Figure 32a with Figure 32b (and Figure 32c with Figure 32d).

Each change of the single construction path from ordinary transmission to special transmission states (or vice versa) requires a pulse sequence of length proportional to the length of the path. For example, each such modification of the upper half of loop $\mathbf{C}_1$ (and of loop $\mathbf{C}_2$) was accomplished with a sequence of length $6n$, where n is the number of cells to be changed. The number n was represented by the loops $\mathbf{C}_1$ and $\mathbf{C}_2$, each of these loops being approximately $2n$ cells long. A delay of $6n$ units was obtained by using a triple return counter with a loop ($\mathbf{C}_1$ or $\mathbf{C}_2$) as its responding organ (Sec. 4.2).

This single path construction procedure could be used for the constructing arm which is operated by the universal constructor. The arm would consist of a single path of transmission cells (sometimes ordinary, sometimes special) going from the universal constructor to the secondary area, and fed by both ordinary and special transmission states. Let ℓ be the number of cells in this single path. To change this path from ordinary to special transmission states (or vice versa), **CU** would send a sequence of length 6ℓ into it. **CU** could determine ℓ from the numbers x_1, y_1, α, β, i, j, which specify the location and size of the secondary automaton as well as the precise place within the secondary where the constructing arm terminates. The numbers x_1, y_1, α, and β are stored as **L** in explicit form, and **CU** could infer i and j by counting the position of λ_{ij} in the sequence of λ's.

Though von Neumann could have used the single path construction procedure for the constructing arm, he actually planned to use a different, and better, procedure. Four pages of rough notes accompanied his manuscript "Theory of Automata: Construction, Reproduction, Homogeneity," which constitutes the present Part II of the present volume. These pages contain his design for a constructing arm and an outline of a program for controlling it. His program is too sketchy to be reconstructed, but his intended design of the construct-

ing arm is plain enough, and once this is known it is not difficult to write a program for it.

Von Neumann's constructing arm is shown in Figures 44–50. Ordinary transmission states are represented by single arrows, and special transmission states are represented by double arrows. This *constructing arm* consists of two adjacent, parallel paths, terminating at a *head* in which one path normally points at the other path. One path is of ordinary transmission states, and the other path is of special transmission states. Thus, both ordinary and special transmission states are always available at the head for the reverse (destruction) process. This being so, it is never necessary to change a whole path from ordinary to special transmission states or vice versa, as in the single path construction procedure.

The constructing arm may be fed by the constructing unit **CU** either from the left (using inputs s and o of Fig. 44) or from below (using inputs s' and o' of Fig. 44). The arm may turn a corner, as in Figure 50.

Von Neumann's procedures for operating the arm, slightly modified, are shown in Figures 44–50. We use a new method of symbolizing pulse sequences in these figures. A pulse sequence which destroys and constructs is represented by a sequence of symbols indicating the quiescent states produced by this pulse sequence.

We will explain this method of symbolizing pulse sequences in connection with the transition from Figures 45a–45b. If the following pulse sequences are fed into s or s' from special transmission or confluent states, they produce the effects indicated.

$$\overline{1110} \text{ changes cell } C1 \text{ from } \mathbf{U} \text{ to } \Downarrow$$
$$\overline{1101} \text{ changes cell } C2 \text{ from } \mathbf{U} \text{ to } \Leftarrow$$
$$\overline{1} \text{ changes cell } B2 \text{ from } \uparrow \text{ to } \mathbf{U}$$
$$\overline{10000} \text{ changes cell } B2 \text{ from } \mathbf{U} \text{ to } \rightarrow.$$

Thus $\overline{11101101110000}$ into s or s' is indicated by

$$\Downarrow \Leftarrow \mathbf{U} \rightarrow \text{ into } s \text{ or } s'.$$

This method of symbolizing pulse sequences must be used with the constraints of von Neumann's transition rule in mind (Ch. 2). For example, the sequence

$$\Downarrow \Uparrow \mathbf{U} \text{ into } s \text{ or } s'$$

is not an allowable sequence for Figure 45b, because one special transmission state cannot kill another special transmission state. However, if the appropriate conditions are satisfied, a single pulse

will change a cell into a **U**. Likewise, if the appropriate conditions are satisfied,

$\overline{10000}$ changes **U** into $\rightarrow$
$\overline{10001}$ changes **U** into $\uparrow$
$\overline{1001}$ changes **U** into $\leftarrow$
$\overline{1010}$ changes **U** into $\downarrow$
$\overline{1011}$ changes **U** into $\Rightarrow$
$\overline{1100}$ changes **U** into $\Uparrow$
$\overline{1101}$ changes **U** into $\Leftarrow$
$\overline{1110}$ changes **U** into $\Downarrow$
$\overline{1111}$ changes **U** into **C**.

Figures 45 and 46 show the procedure for advancing the constructing arm 1 unit, either horizontally or vertically. Figure 47 gives the procedure for withdrawing the arm 1 unit horizontally and leaving the two vacated cells in the desired quiescent states γ and δ. The pulse sequences used for γ and δ depend, of course, on the quiescent states desired. For example, if the variable γ has the value **C**, the sequence fed into Figure 47e becomes

$$\mathbf{U} \Rightarrow \mathbf{U}\ \mathbf{C} \text{ into } s \text{ or } s'.$$

This expression represents

$$\overline{1101111111} \text{ into } s \text{ or } s'.$$

Figure 48 gives the procedure for withdrawing the constructing arm one unit vertically and leaving the vacated cells in the desired quiescent states γ and δ. Figure 49 shows how to inject a starting stimulus into the secondary automaton.

This completes the description of von Neumann's five operations for the constructing arm: horizontal advance, vertical advance, horizontal retreat with γ-δ, vertical retreat with γ-δ, and injection of the starting stimulus. These operations suffice for the construction of any initially quiescent automaton in the first quadrant (i.e., with $x_1 \geqq 0$ and $y_1 \geqq 0$). We will now state an algorithm composed from these operations which achieves this result.

This construction algorithm presupposes that β is an even integer. If β is odd, a row of **U**'s can be added to the secondary to make β even, or the algorithm may be modified slightly. The algorithm also presupposes that the secondary's starting stimulus is to be injected from below into the cell whose center is at $(x_1 + \frac{3}{2}, y_1 + \frac{1}{2})$. Otherwise, the instructions must be modified.

The *algorithm for constructing and starting a secondary automaton* on the plan of Figure 50 is:

(1) The constructing arm is extended from the primary automaton to the upper left-hand corner of the area of the secondary automaton. This requires $x_1 + 2$ horizontal advances and then $y_1 + \beta$ vertical advances.

The constructing arm is now ready to advance to the right of the secondary area and then to retreat, constructing two rows of the secondary as it retreats.

(2) The following sequence of operations is repeated $\beta/2$ times.
   (a) The horizontal advance is repeated $\alpha - 2$ times.
   (b) The horizontal retreat with γ-δ is repeated $\alpha - 2$ times.
   (c) The vertical retreat with γ-δ is repeated twice.

At the end of operation (2) the secondary automaton is complete and may now be started.

(3) A starting stimulus is injected from below into the cell located at $(x_1 + \frac{3}{2}, y_1 + \frac{1}{2})$.

The secondary automaton now begins to operate, and the constructing arm may be withdrawn.

(4) The constructing arm is withdrawn to the primary automaton by y_1 vertical retreats with γ-δ followed by $x_1 + 2$ horizontal retreats with γ-δ, both γ and δ always being the unexcitable state **U**.

This concludes the algorithm. The constructing arm could now be used to construct another secondary automaton.

The pulse sequences which execute this algorithm are injected into inputs s and o of the constructing arm. These pulse sequences are a function of the information stored on the linear array **L**. The way the universal constructor transforms the passive information on **L** into the correct pulse sequences will be explained in Section 5.2.3.

Von Neumann's five constructing arm operations are adequate for construction in the first quadrant, but not for construction in the other quadrants. However, it is not difficult to redesign the head and to program operations for advancing to the left, retreating from the left, advancing down, etc. With these additional operations, the universal constructor could construct a secondary automaton in any quadrant of the plane. This presupposes, of course, that the area for the secondary automaton consists of unexcitable states, that there is a sufficiently wide path of unexcitable states from the universal constructor to this area, and that no other automaton interferes with the construction process.

It is clear from Figures 45–49 that the execution of each of von

Neumann's five constructing arm operations is accomplished by a finite pulse sequence. The longest sequence needed is that for a vertical retreat with γ-δ; if γ and δ are either $\rightarrow$ or $\uparrow$, this operation requires a sequence of length 47. Thus the length of the pulse sequence required for each of these five operations is a constant, independent of the length of the constructing arm. In contrast, von Neumann's one-path construction procedure requires a pulse sequence whose length depends on the length of the path; if ℓ is the length of the path, a sequence of length 5ℓ or greater is required. In this respect, von Neumann's two-path construction procedure is far superior to his one-path construction procedure. This superiority can have a profound effect on the organs supplying the pulse sequences for construction, as we shall now see.

5.2.2 Redesign of the memory control **MC**. Von Neumann's two-path construction procedure can be used for operating the linear array **L**. The new arrangement is shown in Figure 51; it should be compared with the old arrangement of Figure 37.

The path for reading cell x_n starts at input v, extends along row 1, passes through cell x_n, returns along row 4, and terminates at output w. Row 1 is also the ordinary transmission path of the two-path constructing arm. Row 2 is the special transmission path of the constructing arm. The head of the constructing arm consists of cells *C2*, *D1*, and *D2*; it differs slightly from the head of Figure 44a. The stimuli into inputs u and v must come from ordinary transmission states. Note that cells *A1* and *A2* do not affect each other, since a confluent state does not feed a confluent state.

The reading process is exactly the same as before (Secs. 4.1.1 and 4.1.5). The sequence $\overline{10101}$ is injected into v from a pulser $\mathbf{P}(\overline{10101})$. This sequence will travel along row 1, down column D, and into cell x_n. What happens next depends upon whether cell x_n is in state **U**, signifying a "zero," or in state $\downarrow$, signifying a "one."

(1) If cell x_n is in state **U**, the first part $\overline{1010}$ of the sequence changes x_n into $\downarrow$, and the remaining part $\overline{1}$ of the sequence travels back along row 4 and out at exit w.

(2) If cell x_n is in state $\downarrow$, the complete sequence $\overline{10101}$ travels through x_n, back along row 4, and out at exit w.

Thus a $\overline{1}$ at w signifies a "zero," while a $\overline{10101}$ at w signifies a "one." These two sequences may be discriminated by means of the $\overline{1}$ vs. $\overline{10101}$ discriminator (Sec. 3.5 and Fig. 25), as before.

At the end of the reading process, the reading and constructing paths are left as in Figure 51b, with cell x_n in state $\downarrow$. The next step is to lengthen or shorten the reading and construction paths of **L**,

and to leave cell x_n in state **U** (representing a "zero") or state $\downarrow$ (representing a "one"). There are four alternatives:

(L0) Lengthen and leave cell x_n in **U**
(L1) Lengthen and leave cell x_n in $\downarrow$
(S0) Shorten and leave cell x_n in **U**
(S1) Shorten and leave cell x_n in $\downarrow$.

These operations are similar to the operations of horizontal advance (Fig. 45), vertical advance (Fig. 46), horizontal retreat with γ-δ (Fig. 47), and vertical retreat with γ-δ (Fig. 48). We will give the pulse sequences for only one case, that of (L1), i.e., lengthening and leaving cell x_n in $\downarrow$. These are given in Figure 52, where the method of symbolizing sequences is the same as in the earlier figures. For example, starting with the situation of Figure 52a, the sequence 1111011110110111001 of ordinary stimuli into u passes through the confluent state into the row of special transmission states and travels down this row. This sequence changes cell x_n, the cells above and below it, and the cell below $x_n + 1$, into $\Downarrow$, $\Downarrow$, $\Rightarrow$, and $\leftarrow$, respectively. Figure 52b shows the situation produced by this sequence.

Thus this operation (L1) is accomplished by sending a sequence into u, then a sequence into v, another sequence into u, and another sequence into v. Since the absence of a stimulus is represented by "0," this operation can be accomplished by sending simultaneously a sequence into u and a sequence into v; see the example of Figure 14 and Section 2.8.3. These two sequences can be produced by pulsers which feed into u and v and which are stimulated in the proper phase relative to one another (Sec. 3.2.1). The other lengthening, shortening, and writing operations (L0), (S0), and (S1) can be handled similarly.

It is important that each of the operations (L0), (L1), (S0), and (S1) may be accomplished by pulse sequences whose lengths are independent of the lengths of the construction paths, i.e., independent of the index n of the cell x_n which is under scan. In contrast, von Neumann's method of operating the linear array **L** requires pulse sequences whose lengths depend on the length of the construction path (Ch. 4). As a consequence, the memory control for the new method of operating **L** can be much simpler than von Neumann's memory control **MC** (Figs. 37, 39–41). In particular, those aspects of von Neumann's **MC** concerned with obtaining and controlling a delay of $6n$ (where n is the subscript of x_n) are unnecessary in the new method.

Since von Neumann's design of **MC**, as modified in Section 5.1.2,

does work, we will not redesign **MC** here. The read-write-erase unit **RWE** for the new method of operating **L** can be constructed from nine pulsers and a $\overline{1}$ vs. $\overline{10101}$ discriminator, suitably arranged. By recoding and simplifying the control signals which pass between **MC** and the constructing unit **CU**, one can eliminate the read-write-erase control **RWEC**, so that **MC** will consist only of the simplified **RWE** unit and a coded channel.[6]

At this point I should like to speculate on von Neumann's thoughts concerning his design for a self-reproducing automaton at the time he stopped working on the manuscript reproduced as Chapters 1–4 above. His design had turned out to be much more complex than he had anticipated.[7] After developing his two-path construction procedure, he must have realized that it could be used on **L**, and that this would greatly simplify the design of **MC** and of the whole machine. Realizing this, he would have wanted to redesign his self-reproducing automaton along new lines. This redesign would have entailed revising Chapter 3 above and starting Chapter 4 afresh. Von Neumann never found the time to revise and complete the manuscript in this way.

Even though von Neumann's design of the memory control **MC** can be greatly improved, it is nevertheless important. Historically, it constitutes the first proof that a tape unit with unlimited memory can be embedded in his 29-state cellular structure. Moreover, it contains many ingenious design techniques for parallel data processing in this cellular structure.

5.2.3 The constructing unit **CU**. After he had designed the constructing arm, there remained for von Neumann the task of designing the constructing unit **CU** itself. As he recognized, **CU** is a finite automaton which interacts with the tape unit **MC** + **L** and also

[6] The complete design of a memory control for a variant of the new method of operating **L** is given in Thatcher's "Universality in the von Neumann Cellular Model."

[7] In a letter to Miodrag Muntyan of the University of Illinois Press, dated Nov. 4, 1952, von Neumann says of his manuscript:

> I have written so far the first chapter, which amounts to about 40 typewritten pages. . . . I am now working on the second chapter which, I expect, will be somewhat longer, perhaps about twice as long. I will also have a third chapter and possibly a fourth one, but the length of these is still uncertain. Also, when the whole is finished, I will have to go over the entire text once more, and this operation is likely to increase the length some more.

Compare this with von Neumann's letter to Goldstine in Sec. 1.1.2.3 above.

These statements show that after completing Ch. 1 above, von Neumann thought he could develop the design (starting with Ch. 2 above) in a chapter about twice as long as Ch. 1.

performs the function of construction. Hence the same kinds of organs and design principles used in the design of the memory control **MC** (Ch. 4) and an arbitrary finite automaton **FA** (Sec. 5.1.3) can be used to design **CU**. Since von Neumann was a skilled designer and programmer,[8] he undoubtedly saw how to design the constructing unit **CU**, and he may even have had a quite specific design plan in mind.

While it is not appropriate for us to work out the full design here, we will say enough to show that a workable constructing unit **CU** does in fact exist. A complete design for a constructing unit (as well as for a universal constructing machine) is given in James Thatcher's "Universality in the von Neumann Cellular Model." [9]

Von Neumann discussed the process of constructing several secondary automata (Sec. 1.7), but it will suffice here to explain the construction of a single secondary automaton in the first quadrant. The information concerning the location and size of the secondary automaton, as well as a complete description of the secondary automaton, is stored on the linear array **L**, as in Figure 50. First comes a period. Then come the location and size parameters x_1 , y_1 , α, and β, each followed by a comma. Next comes the sequence of λ_{ij}'s describing the secondary automaton cell by cell, for $i = 0, \cdots, \alpha - 1$ and $j = 0, \cdots, \beta - 1$. For the sake of simplicity we assume that the λ_{ij}'s are stored in the order in which they are used by **CU**. The sequence of λ_{ij}'s is terminated by a period, which also marks the end of the information on the tape.

This information may be coded in an alphabet of 14 characters: zero, one, comma, period, and 10 values of the λ_{ij} . These 10 values of the λ_{ij} correspond to the 10 quiescent states $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$ and $\alpha = 0, 1, 2, 3$), $\mathbf{C}_{00}$, and **U**. These 14 characters may be represented by four binary digits (bits). It is convenient to avoid the all zero sequence "0000." It is also convenient to employ a fifth bit position for marking purposes. Thus each character is represented by a total of five bits and is stored in five successive cells of the linear array **L**.

The number x_1 will be represented by a sequence of $x_1 + 1$ of the five-bit characters representing "one," and similarly for the numbers

[8] See pp. 6–15 of the "Introduction" to the present work, as well as Chs. 3 and 4.

[9] See also Codd, "Propagation, Computation, and Construction in Two-Dimensional Spaces." This contains a design for a universal constructing machine in a cellular system in which each cell has four immediate neighbors, but only 8 states, as contrasted with von Neumann's 29 states. Dr. Thatcher and Dr. Codd were acquainted with von Neumann's manuscript "The Theory of Automata: Construction, Reproduction, Homogeneity" in manuscript form.

y_1, α, and β. This method of representation can clearly be improved, but it will simplify our discussion to represent all information in the same alphabet.

Let θ_0, θ_1, θ_2, θ_3, θ_4 be the five bits of a character, with θ_0 being the marker bit. The constructing unit **CU** must be capable of interpreting the sequence θ_1, θ_2, θ_3, θ_4 as a unit. Suppose, for example, that $\mathbf{C}_{00}$ is represented by −1010 on **L**, where the dash indicates the position of the marker bit. If the character λ_{ij} on **L** is −1010, **CU** must put cell (i, j) of the secondary automaton in state $\mathbf{C}_{00}$ at the appropriate stage of the construction process. **CU** will accomplish this by replacing γ or δ in the sequences for horizontal or vertical retreat (Figs. 47 and 48) by the sequence $\overline{1111}$, which is the sequence required by the transition rule (Fig. 10) for constructing $\mathbf{C}_{00}$. Hence **CU** must interpret the tape character −1010 as calling for the sequence $\overline{1111}$ in the appropriate context.

Now the four bits of θ_1, θ_2, θ_3, θ_4 are stored in successive cells x_{n+1}, x_{n+2}, x_{n+3}, x_{n+4}. When **CU** so instructs **MC**, **MC** will read a cell and advance the reading loop to the next cell. The amount of time required for this process is a linear function of the index n. Consequently, the four bits θ_1, θ_2, θ_3, and θ_4 are received by **CU** at widely varying times. In von Neumann's terminology (Sec. 3.1.1), the sequence 1010 comes to **CU** freely timed, and in response **CU** must send the rigidly timed sequence $\overline{1111}$ down the construction arm. The four pulses of $\overline{1111}$ must enter the constructing arm at successive times $(\tau, \tau + 1, \tau + 2, \tau + 3)$, since rigid timing is required by the direct process.

Hence the constructing unit **CU** must be able to convert the freely timed sequence θ_1, θ_2, θ_3, θ_4 (e.g., 1010) into a rigidly timed sequence (e.g., $\overline{1111}$). This may be accomplished in either of two ways.

The first method employs state organs like Figure 43. Associate with the 16 characters $\theta_1\theta_2\theta_3\theta_4$ 30 state organs $\mathbf{SO}_0$, $\mathbf{SO}_1$; $\mathbf{SO}_{00}$, $\cdots$, $\mathbf{SO}_{11}$; $\mathbf{SO}_{0000}$, $\cdots$, $\mathbf{SO}_{1111}$, which are to be interconnected by the coded channel of **CU**. These organs are activated (take control) under the influence of the bits θ_1, θ_2, θ_3, θ_4 in the following manner. When θ_1 is transmitted by **MC** to **CU**, it has this conditional effect: if θ_1 is zero, $\mathbf{SO}_0$ is activated; while if θ_1 is one, $\mathbf{SO}_1$ is activated. Next, the bit θ_2 shifts control from $\mathbf{SO}_0$ or $\mathbf{SO}_1$ to one of the state organs $\mathbf{SO}_{00}$, $\mathbf{SO}_{01}$, $\mathbf{SO}_{10}$, $\mathbf{SO}_{11}$ according to the rule: if θ_2 is zero, $\mathbf{SO}_{\theta_1 0}$ is activated, while if θ_2 is one, $\mathbf{SO}_{\theta_1 1}$ is activated. That is, after **CU** has received the two bits θ_1 and θ_2 from **MC**, the state organ $\mathbf{SO}_{\theta_1\theta_2}$ will be "in control." This process is repeated for the remaining bits θ_3 and θ_4, so that after all bits of $\theta_1\theta_2\theta_3\theta_4$ have been read from **L**, ex-

actly one of the 16 state organs $\mathbf{SO}_{\theta_1\theta_2\theta_3\theta_4}$ of **CU** will be activated. This state organ then controls the selection of a rigidly timed sequence. For example, the state organ $\mathbf{SO}_{1010}$ will cause $\overline{1111}$ to be substituted for γ or δ in one of the constructing sequences of Figures 47 and 48.

The conversion of the freely timed sequence 1010 into the rigidly timed sequence $\overline{1111}$ is a static-dynamic conversion. Our second method of making this conversion employs the static-dynamic converter of Figure 53, together with four state organs $\mathbf{SO}^1$, $\mathbf{SO}^2$, $\mathbf{SO}^3$, and $\mathbf{SO}^4$ to keep track of the four digit positions θ_1, θ_2, θ_3, and θ_4, respectively. The inputs and outputs of the static-dynamic converter and of these four state organs are all connected to the main channel of **CU**. The periodic pulsers of Figure 53 are copies of the alternate periodic pulser $\mathbf{PP}(\bar{\bar{1}})$ of Figure 20. The organs in the left-hand columns of Figure 53 are all decoders which receive their inputs from the main channel. The figure is not, of course, drawn to scale.

The static-dynamic converter and the four state organs $\mathbf{SO}^1$, $\mathbf{SO}^2$, $\mathbf{SO}^3$, $\mathbf{SO}^4$ convert a freely timed sequence θ_1, θ_2, θ_3, θ_4 into the corresponding coded rigidly timed sequence in the following way. At the beginning of this conversion the state organ $\mathbf{SO}^1$ is in control and directs the reading of θ_1. After θ_1 is read by **MC** from **L** it is transmitted to **CU**: a pulse from exit i_1 of **MC** (Fig. 37) signifies that θ_1 is "zero," a pulse from exit i_2 of **MC** signifies that θ_1 is "one." Under the control of $\mathbf{SO}^1$, the pulse signifying "one" is coded into a sequence which is recognized only by the Start_1 decoder of Figure 53 (input d_1) and control is transferred to $\mathbf{SO}^2$. Hence if θ_1 is "one," the $\mathbf{PP}(\bar{\bar{1}})$ for θ_1 is activated, while if θ_1 is "zero," the $\mathbf{PP}(\bar{\bar{1}})$ for θ_1 is left in the inactive state.

Similarly, bit θ_2 is transferred to the $\mathbf{PP}(\bar{\bar{1}})$ for θ_2 by means of state organ $\mathbf{SO}^2$ and the Start_2 decoder of Figure 53. The bits θ_3 and θ_4 are handled in the same way. Hence, after the character $-\theta_1\theta_2\theta_3\theta_4$ has been read by **MC** from **L** and transmitted to **CU**, the periodic pulsers of the static-dynamic converter represent $\theta_1\theta_2\theta_3\theta_4$, the j'th periodic pulser being off or on according to whether θ_j is "zero" or "one" ($j = 1, 2, 3, 4$). The control unit **CU** next directs the conversion of this static information into a rigidly timed sequence.

The static representation of the character $-\theta_1\theta_2\theta_3\theta_4$ is converted into the corresponding dynamic sequence $\overline{\theta_1\theta_2\theta_3\theta_4}$ by a single pulse sequence of the main channel which is recognized by the conversion decoders $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, and $\mathbf{D}_4$ of Figure 53. The operation of each decoder $\mathbf{D}_j$ is as follows, for $j = 1, 2, 3, 4$. $\mathbf{D}_j$ emits a pulse which is delayed in Δ_j and then impinges on the confluent state of column A,

row j. If the **PP** ($\bar{\bar{1}}$) for θ_j is on, this pulse is passed and enters channel B; otherwise, this pulse is blocked and does not enter B. The pulse sequence of the main channel which is recognized by the decoders $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, and $\mathbf{D}_4$ enters these four decoders at different times. But the delays Δ_1, Δ_2, Δ_3, and Δ_4 can be adjusted so that the pulses which do enter channel B are in proper phase. For example, if θ_1, θ_2, θ_3, and θ_4 are all one, the sequence $\overline{1111}$ will be emitted from output g.

Since the all-zero sequence is not used to represent any character, θ_1, θ_2, θ_3, and θ_4 cannot all be zero, and at least one stimulus will be emitted from g. Hence, stimulation of the conversion decoders $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, and $\mathbf{D}_4$ will cause the rigidly timed sequence $\overline{\theta_1\theta_2\theta_3\theta_4}$ to leave the static-dynamic converter at exit g. The periodic pulsers of the static-dynamic converter can then be "cleared" back to the inactive state by stimulating the four stop decoders, whose inputs are labeled f.

This completes the discussion of the two methods for converting a static tape character into a dynamic sequence. The second method requires less apparatus than the first.

We now have a method for reading characters from the tape. At the end of Section 5.2.1 we gave an algorithm for constructing and starting a secondary automaton. This algorithm describes the pulse sequences to be injected into the constructing arm as a function of the information x_1, y_1, α_1, β, λ_{00}, $\cdots$, $\lambda_{\alpha-1,\,\beta-1}$ stored on **L**. The design of the constructing unit **CU** now becomes the task of translating this algorithm into a machine design. We restate this algorithm here, showing how **CU** obtains the necessary information for the construction from the linear array **L**. We assume that initially the reading loop of **L** passes through cell x_0, i.e., through the marker bit of the leftmost period on **L**.

The revised *algorithm for constructing and starting a secondary automaton* (see Fig. 50) is:

(1) **CU** extends the constructing arm from itself to the upper left-hand corner of the area of the secondary automaton. This is accomplished by the following two sub-operations:

   (a) **CU** sends pulse sequences into the constructing arm for $x_1 + 2$ horizontal advances. The pulse sequences for each horizontal advance are given in Figure 45. **CU** senses a character (tally) of x_1 on **L**, sends the sequences of Figure 45, and then moves right to the next character on **L**. When **CU** reads the comma on **L**, it sends pulses for two more horizontal advances, and then goes to execute sub-operation (b).

(b) **CU** sends pulse sequences into the constructing arm for $y_1 + \beta$ vertical advances. It does this as follows. **CU** senses each character (tally) on **L**, sends the sequences of Figure 46 to the constructing arm, and advances to the next character on **L**. When **CU** reaches the comma on **L**, it passes over α to β, sends the sequences of Figure 46 for each character of β, and then returns to the comma preceding α.

The constructing arm is now in position to begin construction. **CU** constructs two rows of the secondary automaton at a time, and must do this $\beta/2$ times. **CU** keeps count of these operations by marking the marker positions of the characters of β.

(2) **CU** senses if β contains any unmarked characters. If β contains no unmarked characters, **CU** unmarks all the characters of β and all the λ's, and then proceeds to operation (3). If β contains an unmarked character, **CU** marks two characters of β and executes sub-operations (a), (b), and (c) in that order.

(a) **CU** repeats the horizontal advance $\alpha - 2$ times. It accomplishes this by passing over the first two tallies of α and then sending the sequences of Figure 45 into the constructing arm for each of the remaining tallies of α.

(b) **CU** repeats the horizontal retreat with γ-δ for $\alpha - 2$ times. It does this by marking two characters of α, and then executing the following operation until all characters of α are marked: mark an unmarked character of α and execute the horizontal retreat with γ-δ of Figure 47.

The horizontal retreat with γ-δ requires the substitution of pulse sequences for γ and for δ according to the states λ_{ij} and $\lambda_{i,j-1}$ to be constructed in the cells at the terminus of the constructing arm. To obtain these λ's, **CU** must find them on **L** and move the reading loop from α to them; later **CU** must return the reading loop to α. In both cases, **CU** can sense the place to stop by means of markers: each time it uses a character of α or a λ, **CU** will mark it. We are assuming that the λ's are placed on **L** from right to left in order of use.

After **CU** has executed the horizontal retreat with γ-δ for $\alpha - 2$ times, it removes the markers from α and proceeds to sub-operation (c).

(c) **CU** repeats the vertical retreat with γ-δ twice, using and marking the next two λ's on **L**.

At the end of operation (2), **CU** has completed the construction of the secondary automaton and proceeds to start it.

(3) **CU** injects the starting stimulus into the secondary automaton by means of the sequences of Figure 49. Exit e of this figure is an input to the cell located at $(x_1 + \frac{3}{2}, y_1 + \frac{1}{2})$, and so this presupposes that the secondary automaton is designed to receive its starting stimulus through the bottom of this cell.

The secondary automaton now begins to operate and **CU** proceeds to withdraw the constructing arm back to itself.

(4) **CU** withdraws the constructing arm by sending sequences into it for y_1 vertical retreats with γ-δ followed by $x_1 + 2$ horizontal retreats with γ-δ. The required sequences are given in Figures 48 and 47, respectively. In every case both γ and δ are to be **U**, which means that single pulses are to be used for each occurrence of γ and of δ. **CU** keeps count of these retreats by means of the numbers y_1 and x_1 on **L**.

This concludes the algorithm for constructing and starting a secondary automaton. Note that at the end of the algorithm the universal constructor **CU** + (**MC** + **L**) is again in its starting state.

Now that this algorithm is formulated, the design of the constructing unit **CU** reduces to the problem of translating this algorithm into a machine design. This can be done by using state organs **SO** like Figure 43, interconnected by a coded channel, with the specific interconnections between each state organ and the other state organs reflecting the content of the algorithm. Control within **CU** is shifted from one state organ **SO** to another under the influence of the information on **L** according to the content of the algorithm. Note that the memory control **MC** operates in a similar manner, with the control organs **CO** playing the role of state organs.

In Section 5.1.3, we noted the resemblance of the constructing unit **CU** to a finite automaton **FA**, and the similarity of von Neumann's universal constructor to Turing's universal computing machine. Let us look at these resemblances more closely.

A universal computing machine M_u has two parts: a tape unit **MC** + **L** and a finite automaton **FA** which interacts with this tape unit. Correspondingly, the universal constructor M_c has the two parts **MC** + **L** and **CU**. The constructing unit **CU** performs two interrelated functions: it interacts with **MC** + **L** and it constructs the secondary automaton whose description is stored on **L**. The processes involved in constructing a secondary automaton are not novel: these processes are already employed in the tape unit **MC** + **L**. More specifically, the reading loop is moved from one cell to another cell on **L** by means of the same kind of construction and destruction steps used in constructing a secondary automaton. Thus the constructing arm of Figures 44–50 is very similar to the construct-

ing arm of Figures 51–52, and either arm could be used in place of the other.

This comparison shows that the constructing unit **CU** is really a special kind of finite automaton, and that the same kinds of organs and design principles used in the design of the memory control **MC** (Ch. 4) and an arbitrary finite automaton **FA** (Sec. 5.1.3) can be used to design **CU**. This comparison shows also that in the context of von Neumann's cellular structure, the output of the universal computing machine M_u is not as different from the output of the universal constructor M_c as it might seem. The output of M_u is a computation, while the output of M_c is a construction, but both are accomplished by sending signals into constructing arms.

This concludes our discussion of the constructing unit **CU**. We have shown that it can be embedded in von Neumann's 29-state cellular structure, and we have stated the general principles of its design.

The constructing unit **CU** together with the tape unit **MC** + **L** constitutes a universal constructor. Hence *there can be embedded in von Neumann's 29-state cellular structure a universal constructor M_c with this property: for each initially quiescent automaton M, there is a coded description $\mathfrak{D}(M)$ of M such that, when $\mathfrak{D}(M)$ is placed on a tape* **L** *attached to M_c, M_c will construct M.*

This answers von Neumann's question about construction universality: can any single automaton construct every other automaton (Sec. 1.1)? The only question remaining is his question about automata self-reproduction: can an automaton construct copies of itself? We will present an affirmative answer to this question also, after first summarizing von Neumann's accomplishments in the present work.

5.3 Conclusion

5.3.1 Summary of the present work. Von Neumann began the present Part II by asking five main questions, some of which contain sub-questions (Sec. 1.1.2.1). The first main question concerns

(A) *Logical universality:*

(A1) When is a class of automata logically universal?

(A2) Is any single automaton logically universal?

A finite automaton with an indefinitely extendible tape is called a Turing machine (see the end of the Second Lecture of Part I above). Turing showed that the class of Turing machines is logically universal in the sense that any logical process (computation) that is at all performable by finite but arbitrarily extensive means can be per-

formed by a Turing machine. Turing further showed that there is a universal Turing machine, that is, a Turing machine which can perform any given computation.

Thus, as von Neumann stated, Turing has answered these two questions about logical universality. Von Neumann then posed analogous questions about construction.

(B) *Constructibility:*
- (B1) Can an automaton be constructed by another automaton?
- (B2) What class of automata can be constructed by a single suitable automaton?

(C) *Construction universality:* Is any single automaton construction universal?

(D) *Self-reproduction:*
- (D1) Is there a self-reproducing automaton?
- (D2) Is there an automaton which can both reproduce itself and perform further tasks?

Von Neumann promised to answer all these questions affirmatively by constructive means, that is, by designing various kinds of constructing and self-reproducing automata.

Questions (C) and (D) lead to his last main question.

(E) *Evolution:*
- (E1) Can the construction of automata by automata progress from simpler types to increasingly complicated types?
- (E2) Assuming some suitable definition of efficiency, can this evolution go from less efficient to more efficient automata?

Von Neumann made a few remarks relevant to evolution in Sections 1.7 and 1.8 but never returned to this topic.[10]

After formulating the five main questions, von Neumann proceeded to make questions (B)–(D) more precise. In effect, the rest of Chapter 1 and all of Chapter 2 are directed to this task. We will give a brief summary of the development of these chapters.

Idealized neurons are adequate to handle the purely logical functions of automata, but construction requires organs which can perform those non-logical functions which are required for the acquisition and combination of the organs of which the constructed automata are composed (Sec. 1.2). In his kinematic model, von Neumann introduced girders, sensing elements, kinematic elements, joining elements, and cutting elements to achieve these non-logical functions

[10] See also our discussion of his "probabilistic model of self-reproduction and evolution" at the end of Sec. 1.1.2.3.

(Fifth Lecture, Part I, and Sec. 1.1.2.3). In the present attempt, he preferred to avoid the kinematical aspects of self-reproduction so that he could concentrate on the logical-combinatorial aspects of this topic (Sec. 1.3.1.1). He was thus led to work with a space (framework) in which the distinction between rest and motion is replaced by a distinction between quiescent states and active states (Secs. 1.3.2 and 1.3.4.1).

Von Neumann then put various restrictions on the space (framework) in which he would carry out his automata constructions. He wanted it to have a high degree of regularity. He required functional homogeneity, though not total homogeneity, because the latter is incompatible with computation and construction (Sec. 1.3.3.2). He further required isotropy, and selected a 2-dimensional space (Sec. 1.3.3.3). Because of the difficulties in modeling automata construction in a continuous space, he decided to work with a discrete space (Sec. 1.3.3.4; cf. Sec. 1.1.2.3). In sum, he decided to carry out his automata designs in a 2-dimensional, regular, cellular structure which is functionally homogeneous and isotropic.

He then decided to model the growth of neurons (excitable cells) by the transformation of existing, unexcitable cells into excitable cells. Such a transformation cannot be induced by ordinary stimuli (the ordinary active states of neurons) since these control the logical functions. Hence, to accomplish construction in his cellular structure von Neumann introduced special stimuli, which cause transitions from the unexcitable state to different species of excitable states. Thus growth is modeled by the transformation of unexcitable cells to excitable cells by special stimuli (Sec. 1.3.4.2). This distinction between ordinary and special stimuli, though modified later (Secs. 1.3.4.3, 2.5, and 2.6), is the basis for an answer to question (B1), Can an automaton be constructed by another automaton?

The next question is (B2), What class of automata can be constructed by a single suitable automaton? Von Neumann referred to the constructing and constructed automata as the "primary" and "secondary" automata, respectively. In Section 1.4 he planned the general organization and mode of operation of a primary automaton which can construct any member of some infinite class of secondary automata. A description of the desired secondary automaton is to be given to the primary automaton. The main problem concerns the exact way this is to be done. Since there is no bound on the size of the secondary to be constructed by a single primary, these descriptions cannot be stored in the primary proper. Working with the universal Turing machine in mind, von Neumann introduced an indefinitely

extendible linear array **L**, on which the description of any secondary, or series of secondaries can be stored.

Thus the primary (constructing) automaton will consist of a finite part plus an indefinitely expandible linear array **L**. A constructing automaton is analogous to a Turing machine, which consists of a finite automaton plus an indefinitely expandible tape. Indeed, as von Neumann noted, the linear array **L** could also serve as an indefinitely expandible tape for a Turing machine if the finite automaton part of a Turing machine could be embedded in the cellular structure (Sec. 1.4.2.3; cf. Sec. 5.1.3). The detailed problem of actually designing, for the cellular structure, finite automata which can interact with the linear array **L** and carry out constructions and computations on the basis of the information obtained from **L**, is solved in Sections 3.1 through 5.2. Thus, by the end of Section 5.2, question (B) has been answered affirmatively. Moreover, question (A), as applied to von Neumann's cellular system, has also been answered affirmatively.

In the balance of Chapter 1, von Neumann reduced questions (C) and (D) to question (B). He reduced question (C) to question (B) by outlining a plan for converting the primary (constructing) automaton into a universal constructor (Sec. 1.5). This plan is illustrated in Figure 50 for constructions in the first quadrant. The secondary automaton is α cells wide and β cells high, and its lower left-hand cell is located at (x_1, y_1). Let ℓ be the number of states that each cell of the secondary is to assume, and let $\lambda = 0, 1, \cdots, \ell - 1$. The state of cell (i, j) is specified by λ_{ij}, where $i = 0, 1, \cdots, \alpha - 1$ and $j = 0, 1, \cdots, \beta - 1$. Hence the plan of any secondary automaton may be given to the universal constructor by placing the sequence $x_1, y_1, \alpha, \beta, \lambda_{00}, \cdots, \lambda_{\alpha-1,\beta-1}$ on the tape **L**. The universal constructor can construct the secondary on the basis of the information contained in this sequence. This reduces question (C) to question (B).

Von Neumann then reduced question (D) to question (C) by showing how to make the universal constructor reproduce itself (Secs. 1.6, 1.7). In essence, he accomplished this by placing a description of the universal constructor on its own tape **L**. He discussed two interrelated points in this connection.

First, there is an apparent difficulty in using **L** for self-reproduction. A self-reproducing automaton must contain a complete description of itself. This might seem a priori impossible, on the ground that the constructing automaton must contain a complete plan of the constructed automaton, and in addition the ability to interpret and execute this plan (Sec. 1.6.1.1; cf. pp. 79–80 of Part I). The difficulty is circumvented by designing the universal constructor so that it uses

the information on **L** twice; once to construct a secondary, and once to make a copy of **L** which is attached to the secondary (Sec. 1.6.1.2; cf. pp. 84–87 of Part I and Sec. 5.3.2 below). In this way, a self-reproducing automaton stores a complete description of itself in a proper part of itself, namely, on the tape **L** (cf. Fig. 55). Likewise, an automaton which is both a universal constructor and a universal computer can store a complete description of itself in a proper part of itself (Fig. 56).[11]

The second point about using **L** in self-reproduction concerns some alternatives. The universal constructor M_c constructs a secondary automaton G whose description $\mathfrak{D}(G)$ is stored on **L**. Might a universal constructor be designed which could directly copy automaton G itself (Sec. 1.6.2.3)? Alternatively, might an automaton be designed which could explore an automaton G and construct its description $\mathfrak{D}(G)$ (Sec. 1.6.3.1)? Von Neumann argued that these alternatives would be difficult, if not impossible, to carry out. In exploring one part of an active automaton G, the exploring automaton might modify an as yet unexplored part of G. More generally, the active G might actively interfere with the exploratory activities of the exploring automaton. This difficulty would be particularly acute in the case of self-reproduction. If a universal constructor were to work directly from the secondary G, when the universal constructor attempted to reproduce itself it would be trying to explore itself. Von Neumann thought such an attempt would probably lead to paradoxes of the Richard type (Sec. 1.6.3.2). None of these difficulties arise when the universal constructor works with the quiescent description $\mathfrak{D}(G)$ (Sec. 1.4.2.1).

There is a parallel problem with respect to the construction of the secondary automaton. If part of the secondary automaton were active during construction, it could interfere with the construction process. Von Neumann solved this problem by stipulating that the initial state of the secondary automaton is to be composed entirely of quiescent states (Sec. 1.7.2.1). In terms of the 29-state transition function developed in Chapter 2, this means that the λ_{ij} are limited to the 10 values **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), and $\mathbf{C}_{00}$. After the secondary automaton has been completed, it may be rendered

[11] We have already noted the parallelism between Turing machines and constructing machines. This parallelsm extends to the present point, for a Turing machine can contain a description of itself. See C. Y. Lee, "A Turing Machine Which Prints its Own Code Script" and James Thatcher's "The Construction of a Self-Describing Machine."

active by a starting stimulus injected into its periphery. We have already called such automata "initially quiescent automata" (Sec. 5.1.2).

Thus the class of automata constructible by the universal constructing automaton is a proper subclass of the automata which can be specified as part of an initial cell assignment (i.e., at time zero) of von Neumann's cellular structure (Sec. 1.3.3.5). Actually, it is possible to design in the 29-state cellular structure constructing automata which could construct every initially quiescent automaton and many active automata as well. There is no need to do this, however, since both the universal constructor and a universal Turing machine can be designed as initially quiescent automata (Secs. 5.3.2, 5.1.3). Moreover, not all automata which can be specified as part of an initial cell assignment of von Neumann's cellular structure can be constructed from within the cellular structure. For example, the configuration of Figure 13b (time zero) when surrounded by a wide band of unexcitable cells **U** cannot be so constructed. A constructing arm can construct the quiescent states $\mathbf{T}_{100}$ and $\mathbf{T}_{020}$, but after it activates them it cannot withdraw from the surrounding area before $\mathbf{T}_{101}$ and $\mathbf{T}_{021}$ kill each other. A simple example of a non-constructible automaton is the 3×3 configuration consisting of the sensitized state $\mathbf{S}_\theta$ surrounded by cells in state $\mathbf{C}_{00}$.[12]

This concludes our summary of Chapter 1. To make specific his questions about automata construction, von Neumann had to choose a particular cellular structure. Chapter 2 is devoted to this task. In Section 2.1.2 he selected a discrete temporal reference frame and decided that the state of a cell at time $t + 1$ will depend only on its own state and the states of its four immediate neighbors at time t. In the remainder of the chapter he developed a set of 29 states and their transition rule. We refer to the result as "von Neumann's 29-state cellular structure." It is summarized in Section 2.8 and Figures 9 and 10.

[12] Thatcher, "Universality in the von Neumann Cellular Model," Sec. 2.3. Moore, "Machine Models of Self-Reproduction," called a configuration which can exist only at time zero a "Garden-of-Eden" configuration. Every Garden-of-Eden configuration is non-constructible, though not conversely. Moore established a necessary condition for Garden-of-Eden configurations to exist in a cellular structure in which information requires at least 1 unit of time to pass from a cell to its neighbors. Myhill, "The Converse of Moore's Garden of Eden Theorem," showed that this condition is also sufficient. This condition is essentially that the cellular structure be non-backwards deterministic in the sense of Burks and Wang, "The Logic of Automata," Sec. 3.3.

We now reformulate questions (A)–(D) so they apply to von Neumann's 29-state cellular structure, at the same time modifying them somewhat.

(A) *Logical universality:* Can an initially quiescent automaton which performs the computations of a universal Turing machine be embedded in von Neumann's 29-state cellular structure?

(B) *Constructibility:* Can an automaton be constructed by another automaton within von Neumann's 29-state cellular structure?

(C) *Construction-universality:* Can there be embedded in von Neumann's 29-state cellular structure a universal constructor M_c with this property: for each initially quiescent automaton M, there is a coded description $\mathfrak{D}(M)$ of M such that, when $\mathfrak{D}(M)$ is placed on a tape **L** attached to M_c, M_c will construct M?

(D) *Self-reproduction:*

(D1) Can a self-reproducing automaton be embedded in von Neumann's 29-state cellular structure?

(D2) Can there be embedded in von Neumann's 29-state cellular structure an automaton which can perform the computations of a universal Turing machine and can also reproduce itself?

All these questions are answered affirmatively in the present work. The answer to the constructibility question (B) is given at the end of Section 2.8.3 and in Figure 14. For each initially quiescent automaton $\mathfrak{A}$, there are two binary (stimuli, no stimuli) sequences which, when fed into inputs i and j of Figure 14, will construct the automaton $\mathfrak{A}$. These two sequences can be produced by two linear arrays of $\mathbf{T}_{00\epsilon}$ cells, with the ϵ's properly chosen. Hence, these two linear arrays, together with the cells in columns A and B of Figure 14, will construct $\mathfrak{A}$. This shows that for each initially quiescent automaton $\mathfrak{A}$ there is an active automaton which will construct it.

Affirmative answers to the remaining questions are based on the constructions of Chapters 3 and 4.

In Chapter 3 von Neumann designed the basic organs to be used. Let $\overline{i^1 \cdots i^n}$ be an arbitrary finite binary sequence, with "1" representing a stimulus and "0" representing the absence of a stimulus. $\overline{\overline{i^1 \cdots i^n}}$ represents an indefinite repetition of the sequence $\overline{i^1 \cdots i^n}$. Von Neumann developed design algorithms for arbitrary pulsers $\mathbf{P}(\overline{i^1 \cdots i^n})$, arbitrary periodic pulsers $\mathbf{PP}(\overline{\overline{i^1 \cdots i^n}})$, and arbitrary decoders $\mathbf{D}(\overline{i^1 \cdots i^n})$. He designed two specific organs: the triple-return counter Φ and the $\overline{1}$ vs. $\overline{\overline{10101}}$ discriminator Ψ. He concluded Chapter 3 with an algorithm which will design an arbitrary coded channel or wire crossing device.

Chapter 4 is devoted to the design of the memory control **MC**. **MC** reads and writes on the tape **L** under the direction of the constructing unit **CU**. A block diagram of **MC** is given in Figure 37; the operation of **MC** is outlined in Section 4.3.1. The basic operation of reading cell x_n of **L**, writing on x_n, and changing the connection to cell x_{n+1} (lengthening) or cell x_{n-1} (shortening) is carried out in two stages. First, when a pulse comes from the output of **CU** labeled "o_1" to the input of **MC** having this same label, **MC** reads the cell x_n by means of the connecting loop $\mathbf{C}_1$. If x_n stores "zero," **MC** sends a pulse to input i_1 of **CU**; while if x_n stores "one," **MC** sends a pulse to input i_2 of **CU**. Second, **CU** sends a pulse to input o_2 or o_3 of **MC** according as "zero" or "one" is to be written in cell x_n; **CU** also sends a pulse to input o_4 or o_5 of **MC** according as the loops $\mathbf{C}_1$ and $\mathbf{C}_2$ are to be lengthened or shortened. **MC** executes these operations, and when they are finished it sends a pulse to input i_3 of **CU** signifying completion.

The parts of the memory control **MC** are: the read-write-erase unit **RWE**; the read-write-erase control **RWEC**; the delay area **W**; the transfer area **Y**; and the coded channel, consisting of the main channel together with **X**, **Z**, $\mathbf{CC}_1$, $\mathbf{CC}_2$, and $\mathbf{CC}_3$.

Von Neumann did not quite complete the design of **MC**, and what he did finish contained many errors. We have corrected all but one error as we have gone along (Ch. 4 and Sec. 5.1.1). This last error is corrected and the design of **MC** is completed in Section 5.1.2. A much improved design of **MC** is suggested in Section 5.2.2.

The memory control **MC**, the indefinite linear array **L**, the connecting loop $\mathbf{C}_1$, and the timing loop $\mathbf{C}_2$ together constitute a tape unit with unlimited memory capacity. Moreover, **MC** is an initially quiescent automaton which is started by a stimulus impinging on its periphery (input o_1). Hence an initially quiescent automaton which performs the functions of a tape unit with unlimited memory capacity can be embedded in von Neumann's 29-state cellular structure.

A Turing machine consists of such a tape unit together with a finite automaton which can interact with this tape unit. In Section 5.1.3 we showed how to simulate an arbitrary finite automaton by an initially quiescent cellular automaton. Combining these results and applying them to the specific case of a universal Turing machine, we obtained a positive answer to von Neumann's question (A): An initially quiescent automaton which performs the computation of a universal Turing machine can be embedded in von Neumann's 29-state cellular structure.

Von Neumann's universal constructor M_c consists of the construct-

ing unit **CU** combined with the tape unit (**MC** + **L**). See Figure 50. The constructing arm is designed in Section 5.2.1, and the design of the constructing unit **CU** is sketched in Section 5.2.3. Hence there can be embedded in von Neumann's 29-state cellular structure a universal constructor M_c with this property: for each initially quiescent automaton M, there is a coded description $\mathfrak{D}(M)$ of M such that, when $\mathfrak{D}(M)$ is placed on a tape **L** attached to M_c, M_c will construct M.

This answers von Neumann's question (C) and brings us to question (D), concerning self-reproduction.

A comparison of the universal computer M_u and the universal constructor M_c shows that in von Neumann's cellular structure, computation and construction are similar activities. Both M_c and M_u are finite data processors which can interact with an indefinitely extendible tape. Suppose the universal computer M_u is designed to write its output answer on a fresh tape **L**. Then both M_u and M_c produce initially quiescent automata. The universal constructor M_c produces rectangular initially quiescent automata based on the 10 states **U**, $\mathbf{T}_{u\alpha 0}$ ($u = 0, 1$; $\alpha = 0, 1, 2, 3$), and $\mathbf{C}_{00}$. The universal computer M_u produces a linear initially quiescent automaton based on the two states **U** and $\mathbf{T}_{030}$ ($\downarrow$).

5.3.2 Self-reproducing automata. Our task is now to convert the universal constructor M_c of Section 5.2.3 and Figure 50 into a self-reproducing automaton.

Note first that the universal constructor M_c is in fact an initially quiescent automaton. Consequently, a description $\mathfrak{D}(M_c)$ can be placed on the tape **L** attached to M_c. When this is done and M_c is started, the complex $M_c + \mathfrak{D}(M_c)$ will produce a copy of M_c as the secondary constructed automaton. This is not yet self-reproduction, however, for the constructed automaton M_c is smaller than the constructing automaton $M_c + \mathfrak{D}(M_c)$.

In this case the constructing automaton is larger and, in a sense, more complex than the constructed automaton because the constructing automaton contains a complete plan $\mathfrak{D}(M_c)$ of the constructed automaton and, in addition, a unit M_c which interprets and executes this plan (Sec. 1.6.1.1, pp. 79–80 of Part I). To obtain a primary automaton which will construct a secondary as large as itself, we make some modifications in the universal constructor M_c (compare Sec. 1.6.1.2 and pp. 84–86 of Part I).

Let the secondary automaton to be constructed consist of an initially quiescent automaton M together with a tape **L** which stores some tape content $\mathfrak{I}(M)$ initially. See Figure 54. Place the following information on the tape **L** of the universal constructor: a period, the

description $\mathfrak{D}(M)$, a second period, the tape content $\mathfrak{I}(M)$, and a third period. As a special case we allow $\mathfrak{I}(M)$ and the third period to be omitted. It is easy for the universal constructor to detect that $\mathfrak{I}(M)$ is omitted since $\mathfrak{I}(M)$ is written in a five-bit code that does not include the all-zero character (Sec. 5.2.3).

Now change M_c into a modified universal constructor $M_c{}^*$ which executes the following three steps. First, $M_c{}^*$ uses $\mathfrak{D}(M)$ to construct M, as before (Secs. 5.2.1 and 5.2.3). Second, $M_c{}^*$ produces a tape **L** attached to M which stores a period, $\mathfrak{I}(M)$, and a second period. If $M_c{}^*$ finds that there is nothing stored on its tape beyond the second period (i.e., $\mathfrak{I}(M)$ is missing), $M_c{}^*$ will then copy a period, $\mathfrak{D}(M)$, and a second period onto the tape attached to M. Third, $M_c{}^*$ will give a starting stimulus to M.

The second of these steps is a simple tape-copying operation. It can be carried out by the constructing unit **CU** and the constructing arm in a way analogous to the way these organs construct M. The coding is different in the two cases, of course. A cell (i, j) of M is described in $\mathfrak{D}(M)$ by a five-bit character which is stored in five successive cells of the tape of $M_c{}^*$. In contrast, each cell of M's tape is the same as the corresponding cell of $M_c{}^*$'s tape.

For each choice of location (x_1, y_1) of a copy of $M_c{}^*$ there is a description $\mathfrak{D}(M_c{}^*)$. Place this description $\mathfrak{D}(M_c{}^*)$ on the tape of $M_c{}^*$ itself. The complex $M_c{}^* + \mathfrak{D}(M_c{}^*)$ will construct $M_c{}^* + \mathfrak{D}(M_c{}^*)$. This is self-reproduction. See Figure 55.

Hence, *a self-reproducing automaton can be embedded in von Neumann's 29-state cellular structure.* This answers question (D1).

Iterated construction and self-reproduction may be achieved by further modification of the universal constructor (Sec. 1.7).

The initially quiescent automaton M_u is a universal Turing machine (Sec. 5.1.3). Place the description $\mathfrak{D}(M_u + M_c{}^*)$ on the tape **L** of $M_c{}^*$. The primary automaton $M_c{}^* + \mathfrak{D}(M_u + M_c{}^*)$ will then construct as secondary the automaton $(M_u + M_c{}^*) + \mathfrak{D}(M_u + M_c{}^*)$. In this case the constructed automaton is larger, and in a sense more complicated, than the constructing automaton.

Next, attach M_u to $M_c{}^*$ as in Figure 56. Place the description $\mathfrak{D}(M_u + M_c{}^*)$ on the tape of $M_u + M_c{}^*$. The automaton $(M_u + M_c{}^*) + \mathfrak{D}(M_u + M_c{}^*)$ will construct $(M_u + M_c{}^*) + \mathfrak{D}(M_u + M_c{}^*)$, and hence is self-reproductive. After it is completed, the secondary $(M_u + M_c{}^*) + \mathfrak{D}(M_u + M_c{}^*)$ can carry out a computation. Alternatively, if M_u is supplied with its own tape, each $(M_u + M_c{}^*) + \mathfrak{D}(M_u + M_c{}^*)$ can compute and construct simultaneously.

Hence, *there can be embedded in von Neumann's 29-state cellular*

structure an automaton which can perform the computations of a universal Turing machine and can also reproduce itself. This answers question (D2).

All of von Neumann's questions about automata construction and computation (Secs. 1.1.2.1 and 5.3.1) have now been answered affirmatively. His 29-state cellular structure is computation-universal, construction-universal, and self-reproductive. In this cellular structure, self-reproduction is a special case of construction, and construction and computation are similar activities.]

BIBLIOGRAPHY

Birkhoff, Garrett. *Hydrodynamics, A Study in Logic, Fact, and Similitude.* Princeton: Princeton University Press, 1950.

—— and John von Neumann. See John von Neumann.

Boltzmann, Ludwig. *Vorlesungen über Gas Theorie.* Two volumes. Leipzig: Johann Barth, 1896 and 1898.

——. *Wissenschaftliche Abhandlungen.* Edited by Fritz Hasenöhrl. Three volumes. Leipzig: Johann Barth, 1909.

Booth, Andrew D. "The Future of Automatic Digital Computers." *Communications of the Association for Computing Machines* 3 (June, 1960) 339–341, 360.

Brainerd, J. G., and T. K. Sharpless. "The ENIAC." *Electrical Engineering* 67 (Feb., 1948) 163–172.

Brillouin, Leon. *Science and Information Theory.* New York: Academic Press, 1956.

Bulletin of the American Mathematical Society. "John von Neumann, 1903–1957." Vol. 64, No. 3, Part 2, May, 1958, 129 pages. This is a memorial issue. Besides the articles by Ulam and Shannon listed separately in this bibliography, it contains articles on von Neumann's work in lattice theory, theory of operators, measure and ergodic theory, quantum theory, and theory of games and mathematical economics.

Burks, Arthur W. "Computation, Behavior, and Structure in Fixed and Growing Automata." *Behavioral Science* 6 (1961) 5–22. Original version and discussion in *Self-Organizing Systems* (edited by Marshall Yovits and Scott Cameron). New York: Pergamon Press, 1960, pp. 282–311, 312–314.

——. "Electronic Computing Circuits of the ENIAC." *Proceedings of the Institute of Radio Engineers* 35 (August, 1947) 756–767.

——. "Programming and the Theory of Automata." Pp. 100–117 of *Computer Programming and Formal Systems,* edited by P. Braffort and D. Hirschberg. Amsterdam: North-Holland Publishing Company, 1963.

——. "Super Electronic Computing Machine." *Electronic Industries* 5 (July, 1946) 62–67, 96.

——. "Toward a Theory of Automata Based on More Realistic Primitive Elements." Pp. 379–385 of *Information Processing 1962, Proceedings of IFIP Congress 62,* edited by C. M. Popplewell. Amsterdam: North-Holland Publishing Company, 1963.

—— and John von Neumann. See John von Neumann.

—— and Hao Wang. "The Logic of Automata." *Journal of the Association for Computing Machinery* 4 (April, 1957) 193–218 and 4 (July, 1957) 279–297. (Reprinted in Wang, Hao, *A Survey of Mathematical Logic,* pp. 175–223. Peking: Science Press, 1962. Distributed by North-Holland Publishing Company, Amsterdam.)

—— and Jesse B. Wright. "Theory of Logical Nets." *Proceedings of the Institute of Radio Engineers* 41 (October, 1953) 1357–1365. Reprinted in *Se-*

quential Machines—Selected Papers, pp. 193–212. Edited by E. F. Moore. Reading, Mass.: Addison Wesley, 1964.

Church, Alonzo. "Applications of Recursive Arithmetic to the Problem of Circuit Synthesis." *Summaries of Talks Presented at the Summer Institute for Symbolic Logic, Cornell University, 1957*. Princeton: Institute for Defense Analysis, 1960.

Codd, Edgar Frank. "Propagation, Computation, and Construction in Two-Dimensional Cellular Spaces." 152 pp. Ph.D. Dissertation, University of Michigan, 1965.

Eccles, J. C. *The Neurophysiological Basis of Mind*. Oxford: Oxford University Press, 1953.

Estrin, Gerald. "The Electronic Computer at the Institute for Advanced Study." *Mathematical Tables and Other Aids to Computation* 7 (April, 1953) 108–114 and frontispiece.

Gödel, Kurt. "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I." *Monatshefte für Mathematik und Physik* 38 (1931) 173–198. (English translation by Elliott Mendelson in *The Undecidable*, edited by Martin Davis, pp. 4–38. Hewlett, New York: Raven Press, 1965.)

——. "On Undecidable Propositions of Formal Mathematical Systems." Mimeographed notes on lectures delivered at the Institute for Advanced Study, Princeton, New Jersey. February–May, 1934. 31 pages. (Reprinted as pp. 39–74 of *The Undecidable*, above.)

——. "Über die Länge der Beweise." *Ergebnisse eines mathematischen Kolloquiums*, Heft 7, pp. 23–24. (English translation by Martin Davis in *The Undecidable*, above, pp. 82–88.)

Goldstine, H. H., and Adele Goldstine. "The Electronic Numerical Integrator and Computer (ENIAC)." *Mathematical Tables and Other Aids to Computation* 2 (July, 1946) 97–110 and frontispiece.

—— and John von Neumann. See John von Neumann.

Goto, Eiichi. "The Parametron, a Digital Computing Element which Utilizes Parametric Oscillation." *Proceedings of the Institute of Radio Engineers* 47 (August, 1959) 1304–1316.

Hamming, R. W. "Error Detecting and Error Correcting Codes." *Bell System Technical Journal* 29 (1950) 147–160.

Hartley, R. V. L. "Transmission of Information." *Bell System Technical Journal* 7 (1928) 535–563.

Holland, J. H. "Concerning Efficient Adaptive Systems." Pp. 215–230 of *Self-Organizing Systems—1962*. Edited by M. C. Yovits, G. T. Jacobi, and G. D. Goldstein. Washington, D. C.: Spartan Books, 1962.

——. "Outline for a Logical Theory of Adaptive Systems." *Journal of the Association for Computing Machinery* 9 (July, 1962) 297–314.

——. "Iterative Circuit Computers." *Proceedings of the 1960 Western Joint Computer Conference*, pp. 259–265. Institute of Radio Engineers, 1960.

——. "Universal Embedding Spaces for Automata." To be published in *Progress in Brain Research*, a festschrift for Norbert Wiener; Amsterdam: Elsevier Publishing Company.

——. "A Universal Computer Capable of Executing an Arbitrary Number of Sub-Programs Simultaneously." *Proceedings of the 1959 Eastern Joint Computer Conference*, pp. 108–113. Institute of Radio Engineers, 1959.

Kemeny, John. "Man Viewed as a Machine." *Scientific American* 192 (April,

1955) 58–67. This is based on von Neumann's Vanuxem Lectures, delivered at Princeton University in March, 1953.

Keynes, John Maynard. *A Treatise on Probability*. London: Macmillan and Co., 1921.

Kleene, S. C. *Introduction to Metamathematics*. New York: Van Nostrand, 1952.

——. "Representation of Events in Nerve Nets and Finite Automata." Pp. 3–41 of *Automata Studies*, edited by C. E. Shannon and J. McCarthy. Princeton: Princeton University Press, 1956. (This appeared originally as Rand Corporation Memorandum RM-704, 101 pp., dated December 15, 1951.)

Laplace, Marquis Pierre Simòn de. *A Philosophical Essay on Probabilities*. Translated by F. W. Truscott and F. L. Emory. New York: Dover Publications, 1951. First French edition 1814.

Lee, C. Y. "A Turing Machine Which Prints Its Own Code Script." Pp. 155–164 of *Proceedings of the Symposium on Mathematical Theory of Automata, New York, April, 1962*. Brooklyn, New York: Polytechnic Press, 1963.

Metropolis, N., and Ulam, S. "The Monte Carlo Method." *Journal of the American Statistical Association* 44 (1949) 335–41.

McCulloch, W. S., and W. Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics* 5 (1943); 115–133.

McNaughton, Robert. "On Nets Made up of Badly Timed Elements, Part I; Slow but Perfectly Timed Elements." Mimeographed, 30 pages. Philadelphia: Moore School of Electrical Engineering, University of Pennsylvania, 1961.

Moore, E. F. "Machine Models of Self-Reproduction." Pp. 17–33 of *Mathematical Problems in the Biological Sciences*. Proceedings of Symposia in Applied Mathematics, Vol. XIV. Providence, Rhode Island: American Mathematical Society, 1962.

Morgenstern, Oskar, and John von Neumann. See John von Neumann.

Myhill, John. "The Converse of Moore's Garden-of-Eden Theorem." *Proceedings of the American Mathematical Society* 14 (August, 1963) 685–686.

Nyquist, H. "Certain Factors Affecting Telegraph Speed." *Bell System Technical Journal* 3 (1924) 324–346.

Rajchman, J. A. "The Selectron—A Tube for Selective Electrostatic Storage." *Mathematical Tables and Other Aids to Computation* 2 (October, 1947) 359–361 and frontispiece.

Richard, Jules. "Les principes des mathématiques et le problème des ensembles." *Revue générale des sciences pures et appliquées* 16 (1905) 541–543.

Russell, Bertrand. "Mathematical Logic as Based on the Theory of Types." *American Journal of Mathematics* 30 (1908) 222–262.

Shannon, C. E. "A Mathematical Theory of Communication." *Bell System Technical Journal* 27 (1948) 379–423, 623–656. (Reprinted in C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, pp. 3–91. Urbana: University of Illinois Press, 1949.)

——. "Von Neumann's Contributions to Automata Theory." *Bulletin of the American Mathematical Society*, Vol. 64, No. 3, Part 2, May, 1958, pp. 123–129.

Szilard, L. "Über die Entropieverminderung in einem thermodynamischen

System bei Eingriffen intelligenter Wesen." *Zeitschrift für Physik* 53 (1929) 840–856. (English translation by Anatol Rapoport and Mechthilde Knoller, "On the Decrease of Entropy in a Thermodynamic System by the Intervention of Intelligent Beings." *Behavioral Science* 9 (October, 1964) 301–310.)

Tarski, Alfred. "The Concept of Truth in Formalized Languages." Pp. 152–278 of *Logic, Semantics, Metamathematics* (translated by J. H. Woodger); Oxford: Oxford University Press, 1956. (Original Polish version 1933, German translation 1936.)

Thatcher, James. "The Construction of a Self-Describing Turing Machine." Pp. 165–171 of *Proceedings of the Symposium on Mathematical Theory of Automata, New York, April, 1962*. Brooklyn, New York: Polytechnic Press, 1963.

——. "Universality in the von Neumann Cellular Model." 100 pp. Technical Report 03105-30-T, ORA, University of Michigan, 1965. To be published in *Essays on Cellular Automata*, edited by A. W. Burks.

Turing, A. M. "The Chemical Basis of Morphogenesis." *Philosophical Transactions of the Royal Society of London.* Series B, Biological Sciences, Vol. 237, August, 1952, pp. 37–72.

——. "Computability and λ-Definability." *The Journal of Symbolic Logic* 2 (Dec., 1937) 153–163.

——. "On Computable Numbers, With an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society*, Series 2, 42 (1936–37) 230–265. "A Correction," *Ibid.*, 43 (1937) 544–546. (Reprinted in *The Undecidable*, edited by Martin Davis, pp. 115–154, Hewlett, New York: Raven Press, 1965.)

Ulam, S. M. *A Collection of Mathematical Problems.* New York, Interscience Publishers, Inc., 1960.

——. "Electronic Computers and Scientific Research." Pp. 95–108 of *The Age of Electronics.* Edited by C. F. J. Overhage. New York: McGraw-Hill, 1962.

——. "John von Neumann, 1903–1957." *Bulletin of the American Mathematical Society*, Vol. 64, No. 3, Part 2, May, 1958, pp. 1–49. This contains valuable biographical information, a summary account of von Neumann's accomplishments, and a bibliography.

——. "On Some Mathematical Problems Connected with Patterns of Growth of Figures." Pp. 215–224 of *Mathematical Problems in the Biological Sciences.* Proceedings of Symposia in Applied Mathematics, Vol. 14, Providence, Rhode Island: American Mathematical Society, 1962.

——. "Random Processes and Transformations." *Proceedings of the International Congress of Mathematicians*, 1950, Vol. II, pp. 264–275. Providence, Rhode Island: American Mathematical Society, 1952.

John von Neumann—Collected Works. Six volumes. Edited by A. H. Taub. New York: Macmillan, 1961–63. Vol. V is entitled *Design of Computers, Theory of Automata and Numerical Analysis.* References to these volumes will be made as follows: "*Collected Works* 5.288–328" refers to pp. 288–328 of Vol. V.

von Neumann, John. *The Computer and the Brain.* With a preface by Klara von Neumann, pp. v–x. New Haven: Yale University Press, 1958. This manuscript was prepared during 1955 and 1956 for the Silliman Lectures

of Yale University, but von Neumann was never able to deliver the lectures and never completed the manuscript.

——. *First Draft of a Report on the EDVAC*. Contract No. W-670-ORD-4926, between the United States Army Ordnance Department and the University of Pennsylvania. Pp. x + 101 (double spaced) with 22 figures and a table explaining the order code. Mimeographed. Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945. An earlier typed version had more material. Both versions are incomplete, lacking many cross references and terminating with the exposition of the order code. But the logical design is essentially complete except for the control and the input and output organs.

——. "The General and Logical Theory of Automata." Pp. 1–41 in *Cerebral Mechanisms in Behavior—The Hixon Symposium*, edited by L. A. Jeffress. New York: John Wiley, 1951. *Collected Works* 5.288–328. The paper was read on September 20, 1948. Von Neumann's comments on other papers are on pp. 58–63, 96, 109–111, 132, and 232 of *Cerebral Mechanisms in Behavior*.

——. *Mathematische Grundlagen der Quantenmechanik*. Berlin: Springer, 1932. (English translation by Robert Beyer, Princeton University Press, 1955.)

——. "Non-Linear Capacitance or Inductance Switching, Amplifying and Memory Devices." *Collected Works* 5.379–419. This is the basic paper for United States Patent 2,815,488, filed April 28, 1954, issued December 3, 1957.

——. "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components." Pp. 43–98 of *Automata Studies*, edited by C. E. Shannon and J. McCarthy. Princeton: Princeton University Press, 1956. *Collected Works* 5.329–378. This paper is based on notes taken on five lectures given by von Neumann at the California Institute of Technology in January, 1952, which were issued in a mimeographed edition.

——. "Quantum Logics (Strict- and Probability- Logics)." Written in about 1937. *Collected Works* 4.195–197.

——. Remarks in the discussion of Ralph Gerard's "Some of the Problems Concerning Digital Notions in the Central Nervous System." In *Cybernetics*, edited by H. von Foerster, pp. 19–31. New York, Josiah Macy, Jr. Foundation, 1951. This is the transactions of a conference held in March, 1950.

——. Review of Norbert Wiener's *Cybernetics, or Control and Communication in the Animal and the Machine*. *Physics Today* 2 (1949) 33–34.

——. "Zur Theorie der Gesellschaftspiele," *Mathematische Annalen* 100 (1928) 295–320. *Collected Works* 6.1–26.

——, and Garrett Birkhoff. "The Logic of Quantum Mechanics." *Annals of Mathematics* 37 (1936) 823–843. *Collected Works* 4.105–125.

——, Arthur Burks, and H. H. Goldstine. *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. Part I of a report on the mathematical and logical aspects of an electronic computing instrument prepared on Contract W-36-034-ORD-7481 for the Ordnance Department of the United States Army. 42 pp. Princeton: Institute for Advanced Study, 1946. Second edition, 1947, pp. vi + 42. *Collected Works*, 5.34–79.

——, and H. H. Goldstine. *Planning and Coding of Problems for an Electronic*

Computing Instrument. Part II of a report on the mathematical and logical aspects of an electronic computing instrument prepared on Contract W-36-034-ORD-7481 for the Ordnance Department of the United States Army. Princeton: Institute for Advanced Study. Vol. 1, pp. iv + 69, 1947. Vol. 2, pp. iv + 68, 1948. Vol. 3, pp. iii + 23, 1948. *Collected Works* 5.80–235.

—— and Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton: Princeton University Press, 1944. 2nd ed., Princeton, 1947; 3rd ed., Princeton, 1953.

Wiener, Norbert. *Cybernetics, or Control and Communication in the Animal and the Machine.* New York: John Wiley & Sons, Inc., 1948. 2nd ed., with additions, New York, 1961.

Wigington, R. L. "A New Concept in Computing." *Proceedings of the Institute of Radio Engineers* 47 (April, 1959) 516–523.

Wilkes, M. V. *Automatic Digital Computers.* London: Methuen & Co., Ltd., 1956.

——. "Progress in High-Speed Calculating Machine Design." *Nature* 164 (August 27, 1949) 341–343.

Williams, F. C. "A Cathode-Ray Tube Digit Store." *Proceedings of the Royal Society of London,* Series A, Vol. 195 (22 December 1948) 279–284.

FIGURES

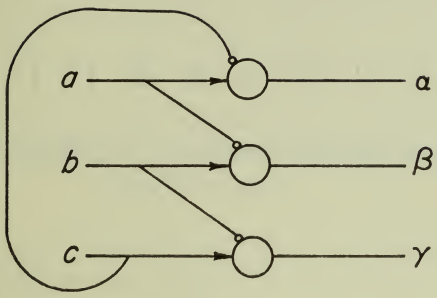


Fig. 1. Neural network in which dominance is not transitive

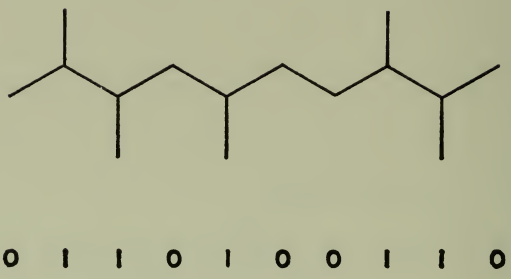


Fig. 2. A binary tape constructed from rigid elements

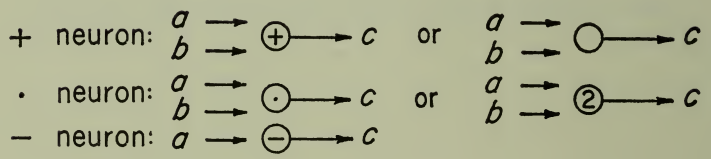
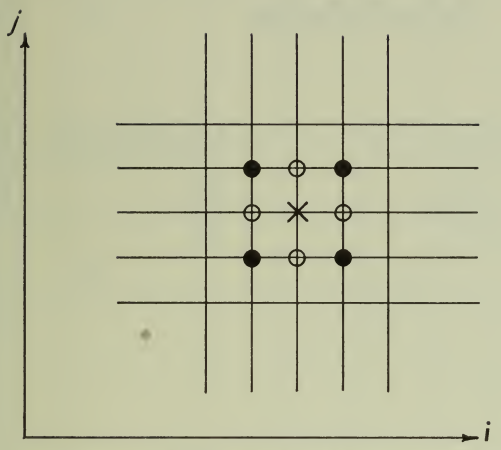
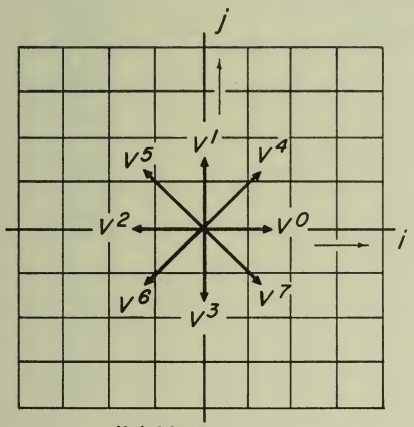


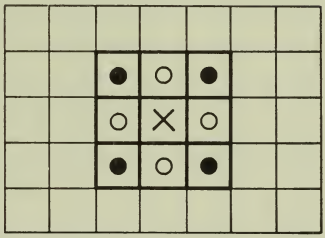
Fig. 3. The basic neurons



(a) Nearest (○) and next nearest (●) neighbors of X



(b) Unit vectors



(c) Nearest (○) and next nearest (●) neighbors of X

Fig. 4. The quadratic lattice

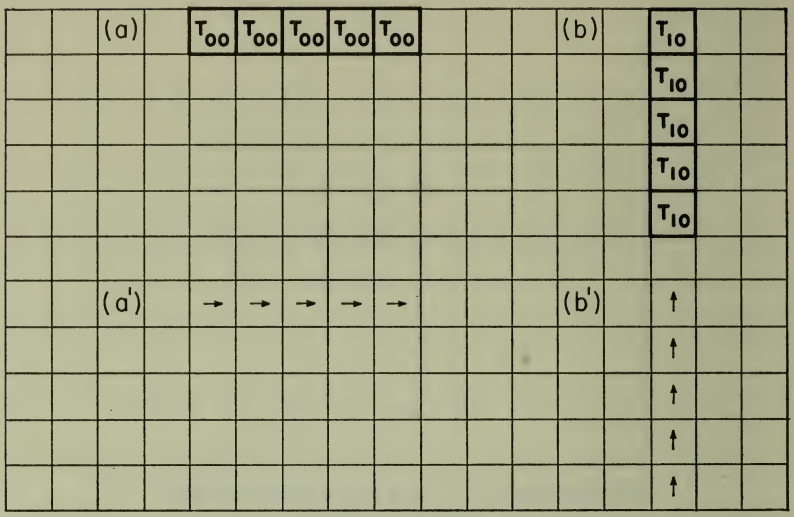
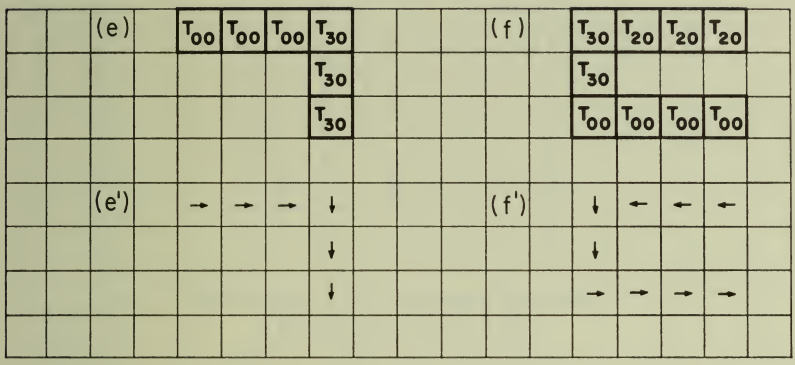


Fig. 5. Ordinary transmission states



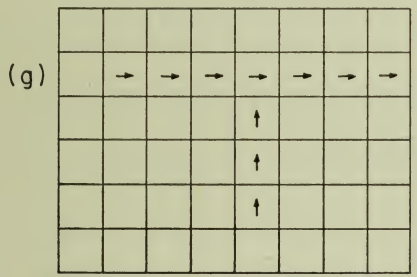
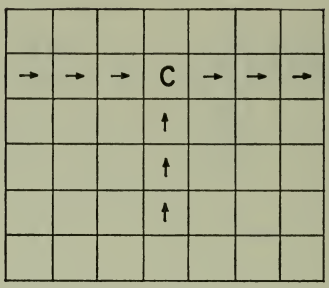
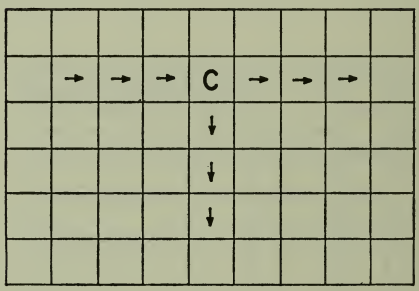


Fig. 5. Part two



(a) Achievement of a · neuron by using confluent states



(b) Achievement of a split line by using confluent states

Fig. 6. Confluent states

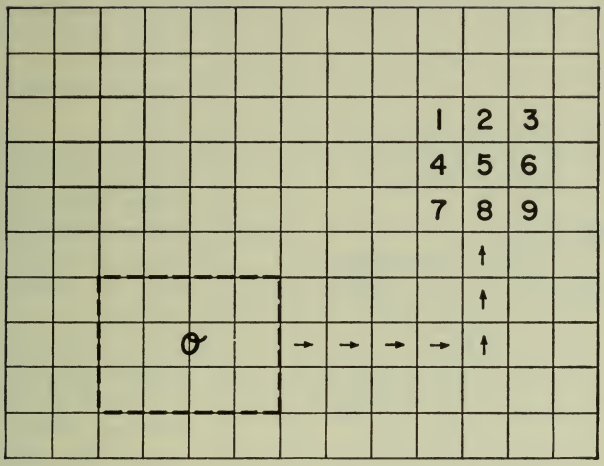


Fig. 7. Organization of an area

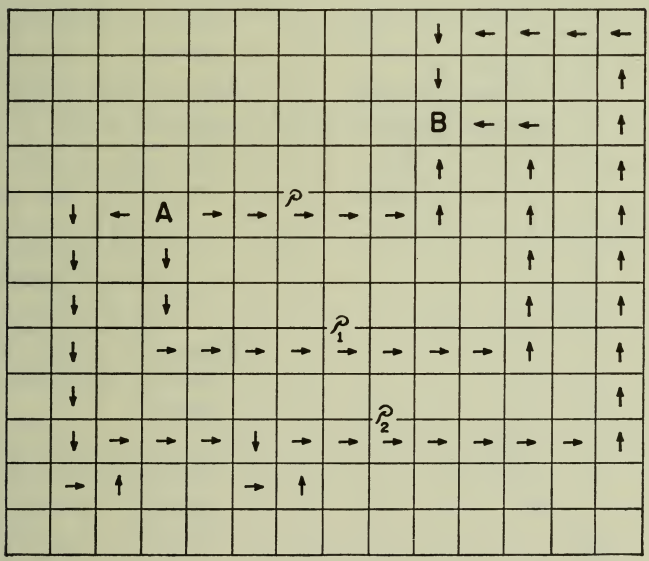


Fig. 8. The need for both even and odd delays

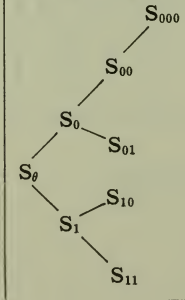
Class	Name	Symbol	Number	Summary of transition rule
	Unexcitable	$\mathbf{U}$	1	Direct process changes **U** into sensitized states and then into $\mathbf{T}_{u\alpha 0}$ or $\mathbf{C}_{00}$. Reverse process kills $\mathbf{T}_{u\alpha\epsilon}$ or $\mathbf{C}_{\epsilon\epsilon'}$ into **U**.
Ordinary	Confluent	$\mathbf{C}_{\epsilon\epsilon'}$ $\begin{cases}\epsilon = 0 \text{ (quiescent)}\\ \epsilon = 1 \text{ (excited)}\end{cases}$ $\begin{cases}\epsilon' = 0 \text{ (next quiescent)}\\ \epsilon' = 1 \text{ (next excited)}\end{cases}$	4	Receives conjunctively from $\mathbf{T}_{0\alpha\epsilon}$ directed toward it; emits with double delay to all $\mathbf{T}_{u\alpha\epsilon}$ not directed toward it. Killed to **U** by $\mathbf{T}_{1\alpha 1}$ directed toward it; killing dominates reception.
Special	Transmission ($\mathbf{T}_{u\alpha\epsilon}$)	$\mathbf{T}_{0\alpha\epsilon}$: $\mathbf{0}\rightarrow$, $\uparrow\mathbf{0}$, $\mathbf{0}\leftarrow$, $\downarrow\mathbf{0}$ *Class:* $u = 0$ (ordinary), $u = 1$ (special) *Excitation:* $\epsilon = 0$ (quiescent), $\epsilon = 1$ (excited)	8	Receives disjunctively from $\mathbf{T}_{0\alpha\epsilon}$ directed toward it and from $\mathbf{C}_{\epsilon\epsilon'}$; emits in output direction with single delay (a) to $\mathbf{T}_{0\alpha\epsilon}$ not directed toward it and to $\mathbf{C}_{\epsilon\epsilon'}$ (b) to **U** or sensitized states by direct process (c) to kill $\mathbf{T}_{1\alpha\epsilon}$ by reverse process. Killed to **U** by $\mathbf{T}_{1\alpha 1}$ directed toward it; killing dominates reception.
		$\mathbf{T}_{1\alpha\epsilon}$: $\mathbf{1}\rightarrow$, $\uparrow\mathbf{1}$, $\mathbf{1}\leftarrow$, $\downarrow\mathbf{1}$ *Output direction:* $\alpha = 0$ (right), $\alpha = 1$ (up), $\alpha = 2$ (left), $\alpha = 3$ (down)	8	Receives disjunctively from $\mathbf{T}_{1\alpha\epsilon}$ directed toward it and from $\mathbf{C}_{\epsilon\epsilon'}$; emits in output direction with single delay (a) to $\mathbf{T}_{1\alpha\epsilon}$ not directed toward it (b) to **U** or sensitized states by direct process (c) to kill $\mathbf{T}_{0\alpha\epsilon}$ or $\mathbf{C}_{\epsilon\epsilon'}$ by by reverse process. Killed to **U** by $\mathbf{T}_{0\alpha 1}$ directed toward it; killing dominates reception.
	Sensitized		8	These are intermediary states in the direct process. $\mathbf{T}_{u\alpha 1}$ directed toward **U** converts it to S_θ. Thereafter, S_Σ is followed by (a) $S_{\Sigma 1}$ if some $\mathbf{T}_{u\alpha 1}$ is directed toward the cell (b) $S_{\Sigma 0}$ otherwise, until the direct process terminates in a $\mathbf{T}_{u\alpha 0}$ or $\mathbf{C}_{00}$. See Figure 10.

Fig. 9. The 29 states and their transition rule

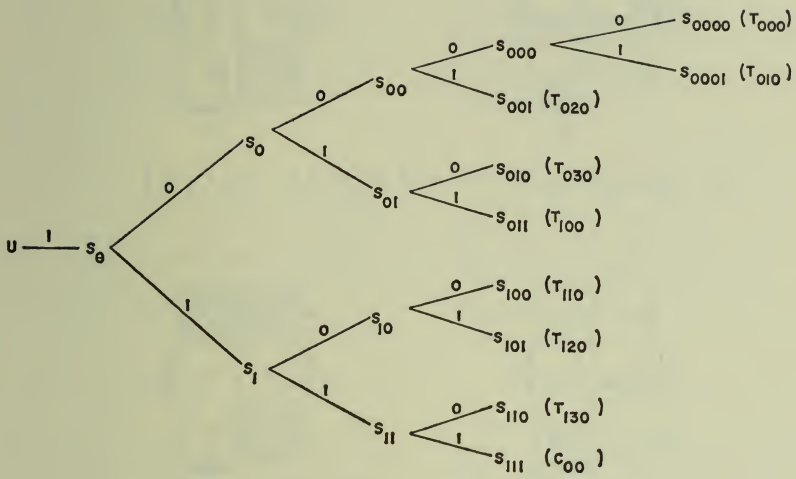
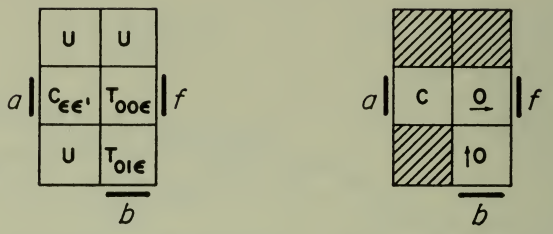
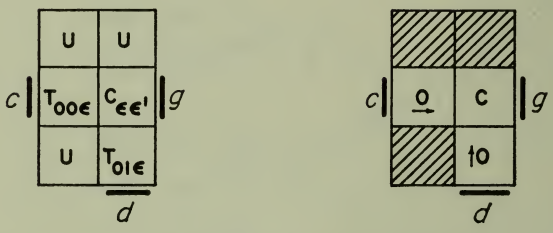


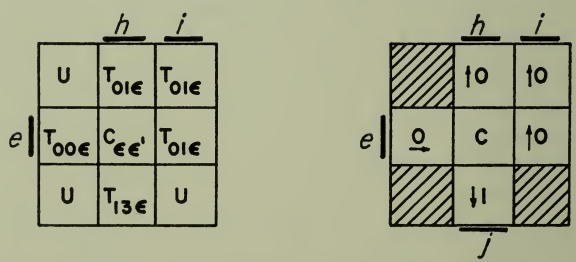
Fig. 10. Succession of states in the direct process



(a) Realization of $f(t+3) = [a(t) + b(t+1)]$



(b) Realization of $g(t+3) = [c(t) \cdot d(t)]$



(c) Conversion of ordinary stimuli into special stimuli by confluent states, and wire-branching

Fig. 11. Illustrations of transmission and confluent states

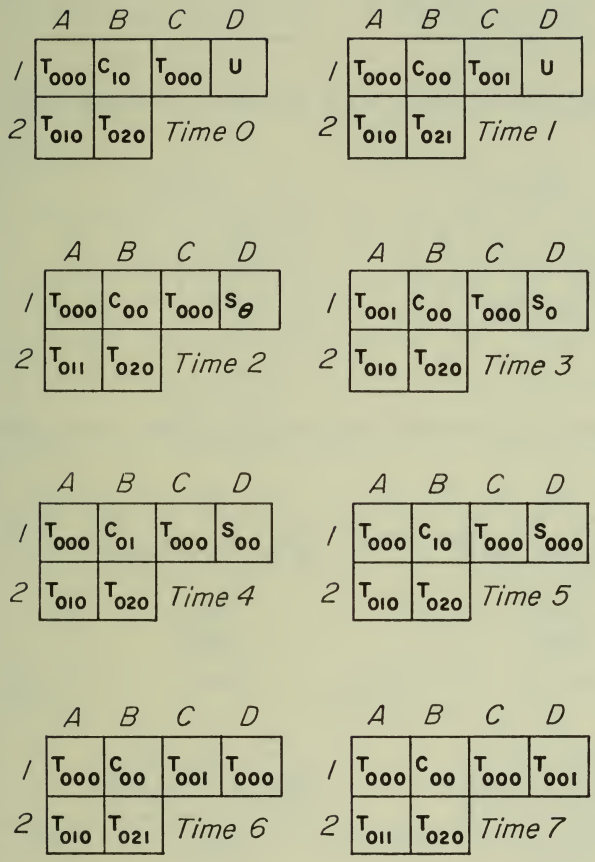
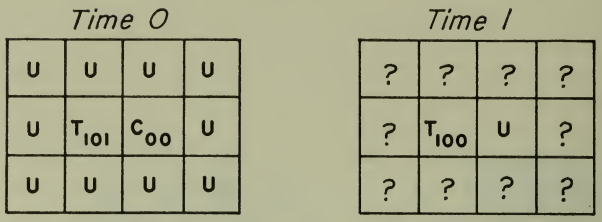
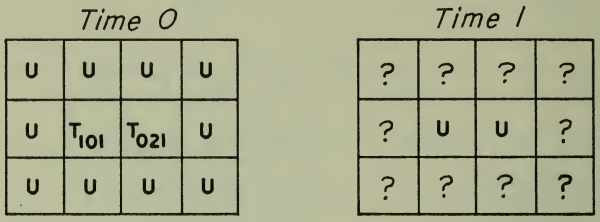


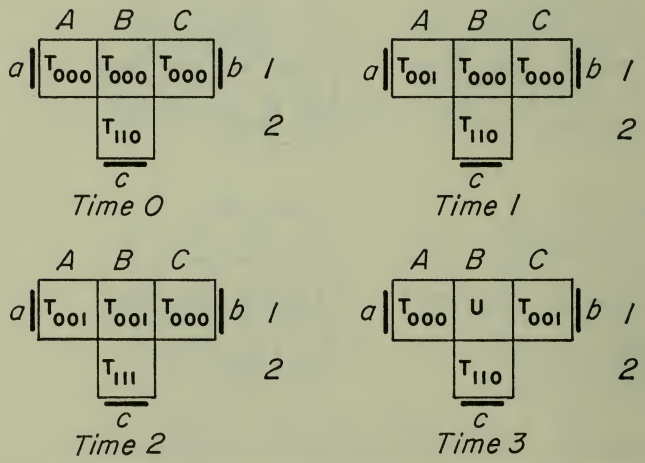
Fig. 12. Illustration of the direct process



(a) Special transmission state killing a confluent state.



(b) Special and ordinary transmission states killing each other.



(c) Killing dominates reception but not emission.

Fig. 13. Examples of the reverse process

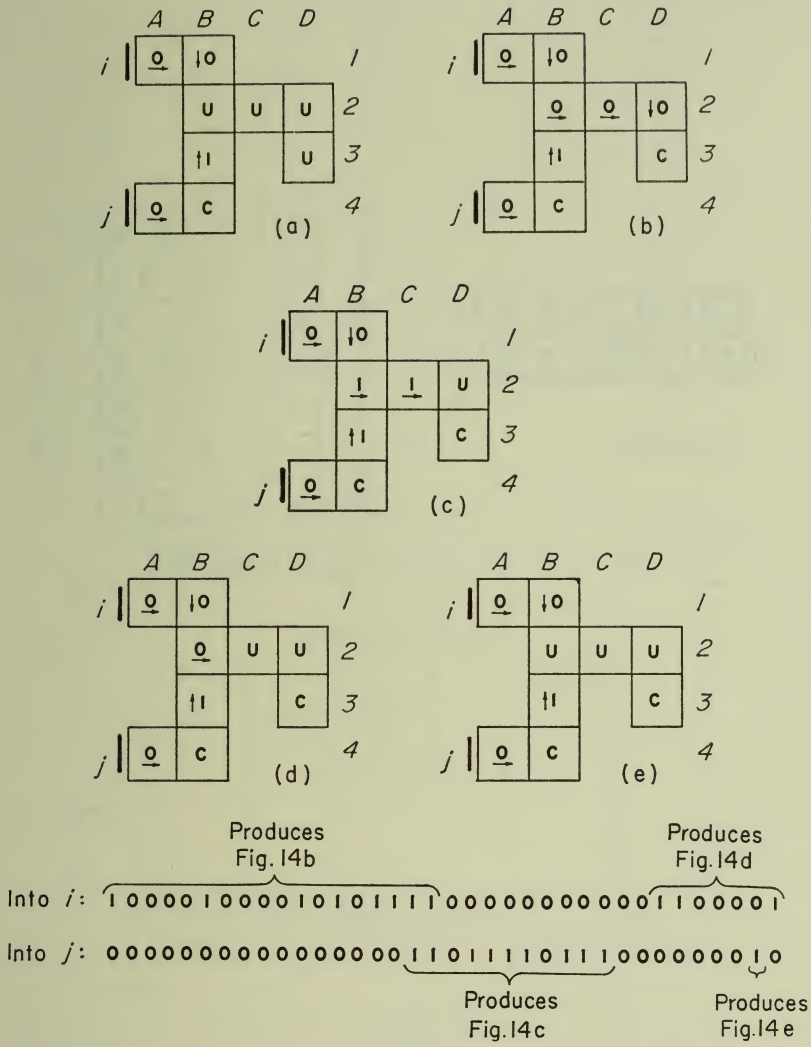


Fig. 14. Procedure for modifying a remote cell and returning the constructing path to its original unexcitable state

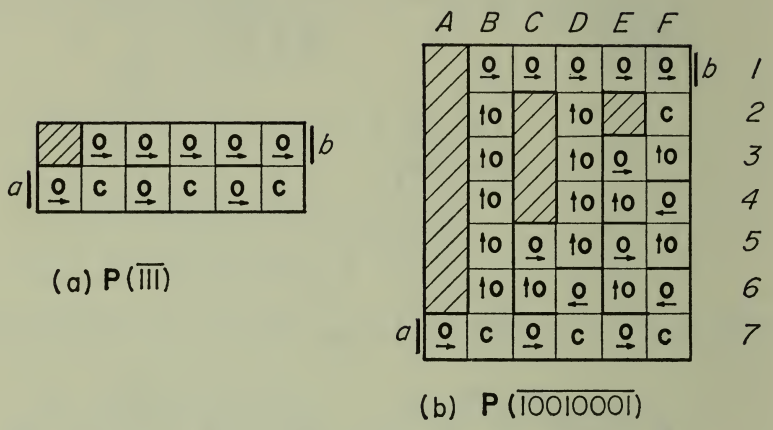
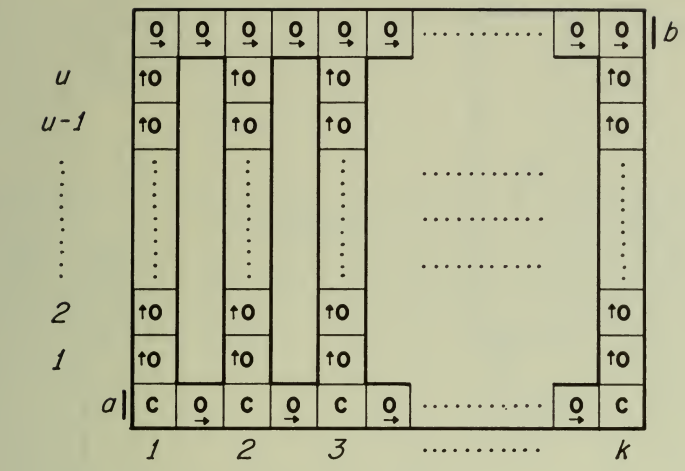
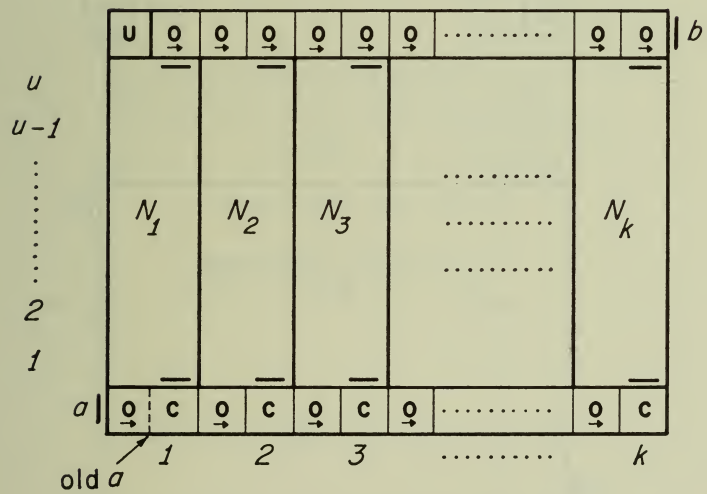


Fig. 15. Two pulsers

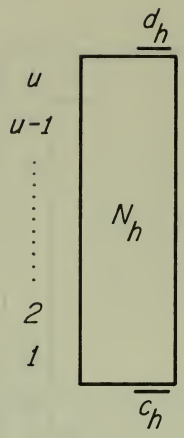


(a)

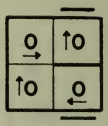


(b)

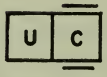
Fig. 16. Design of a pulser



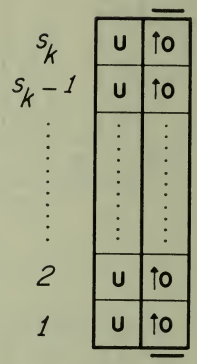
(c)



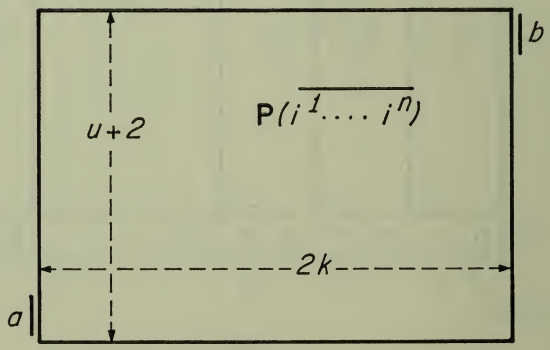
(d)



(e)

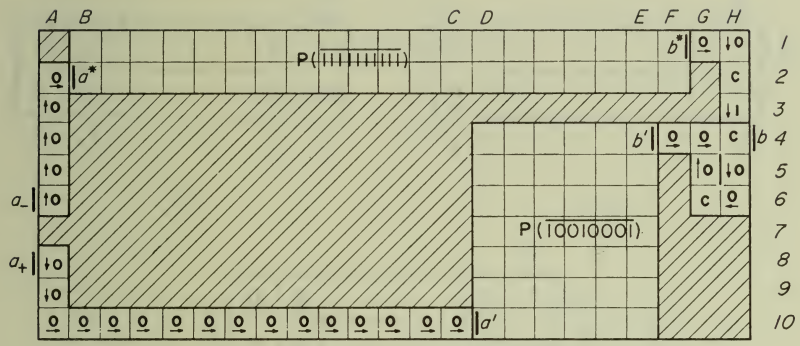


(f)

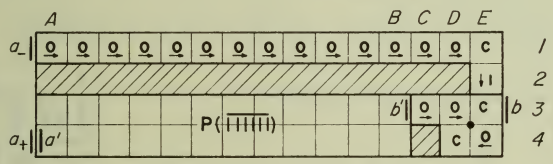


(g)

Fig. 16. Part two



(a) Periodic Pulser **PP** $(\overline{1001000 1})$



(b) Special Periodic Pulser **PP** $(\bar{1})$

Fig. 17. Two periodic pulsers

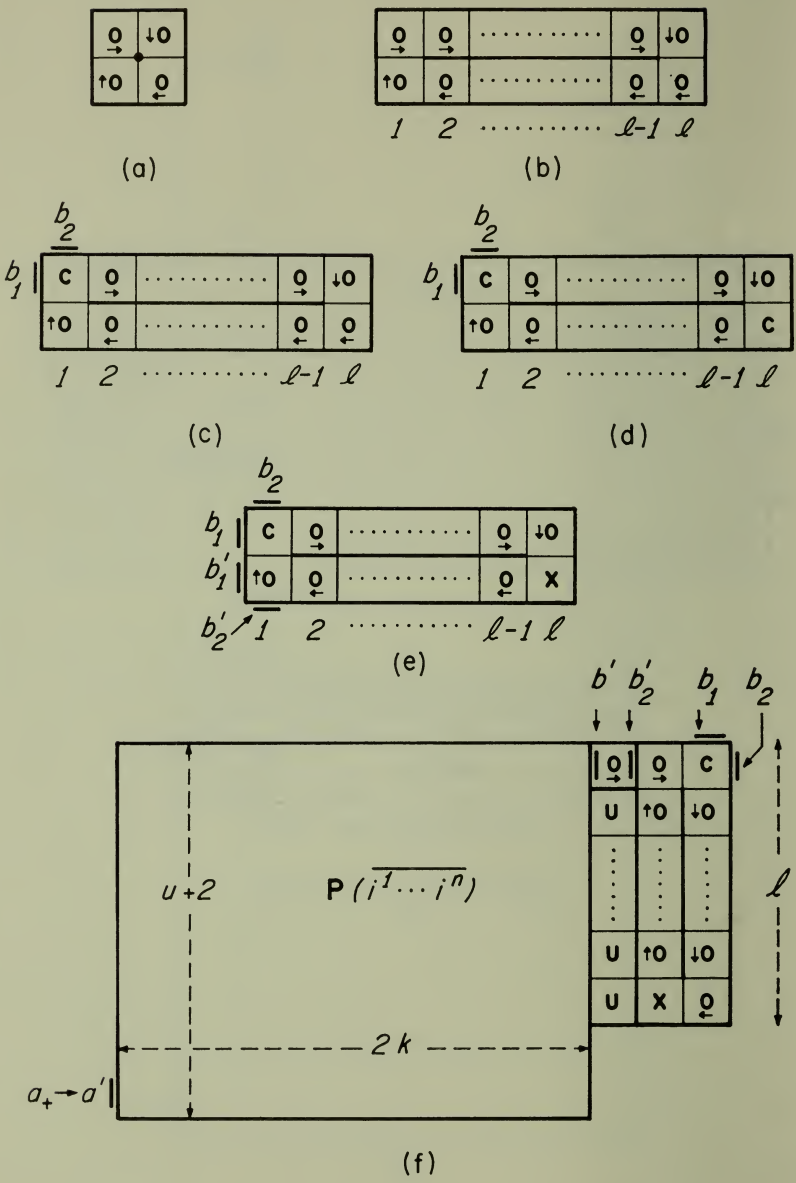
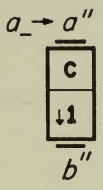
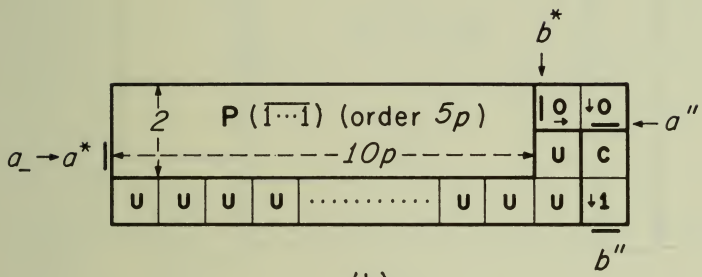


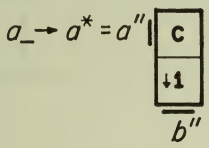
Fig. 18. Design of a periodic pulser: initial construction



(g)



(h)



(i)

Fig. 18. Part two

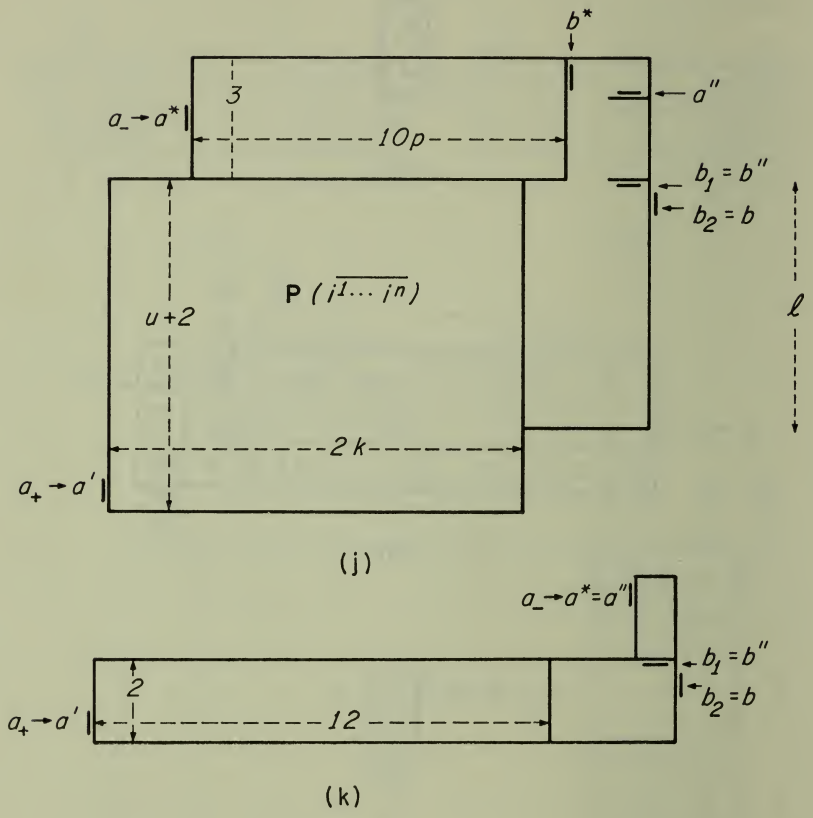
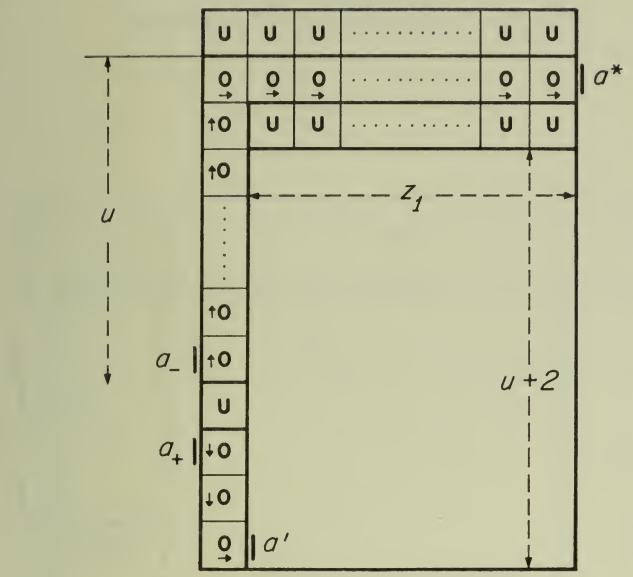
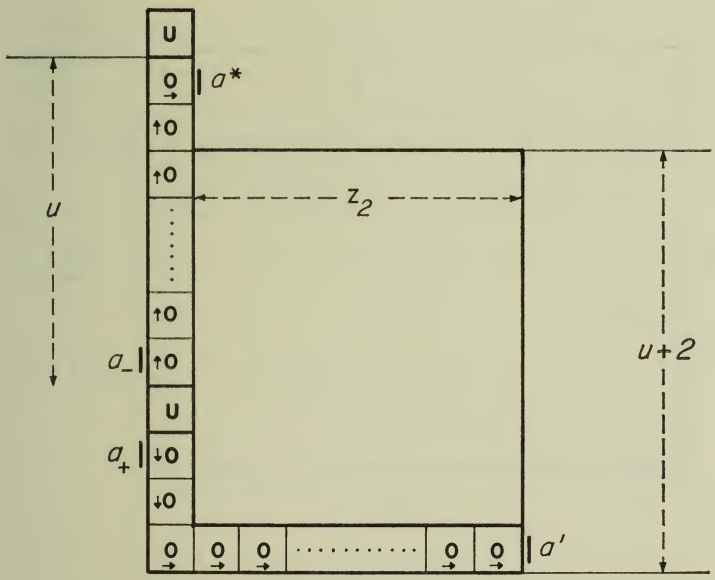


Fig. 18. Part three

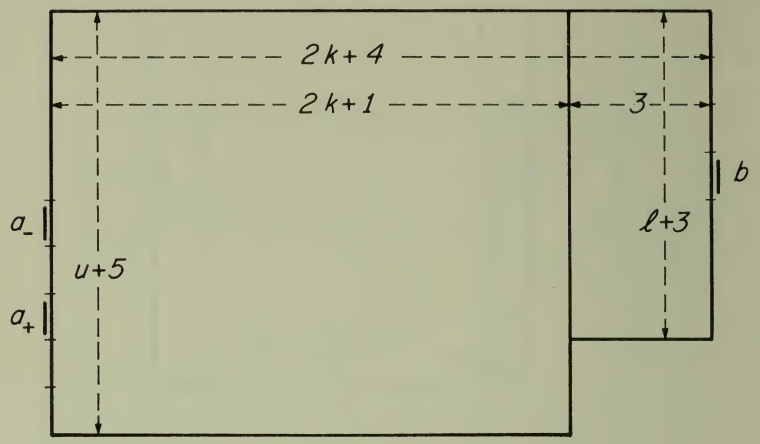


(a)

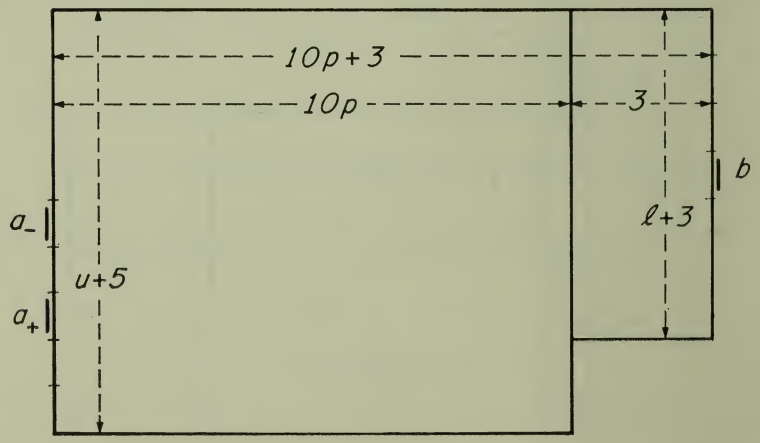


(b)

Fig. 19. Design of a periodic pulser: equalization of delays

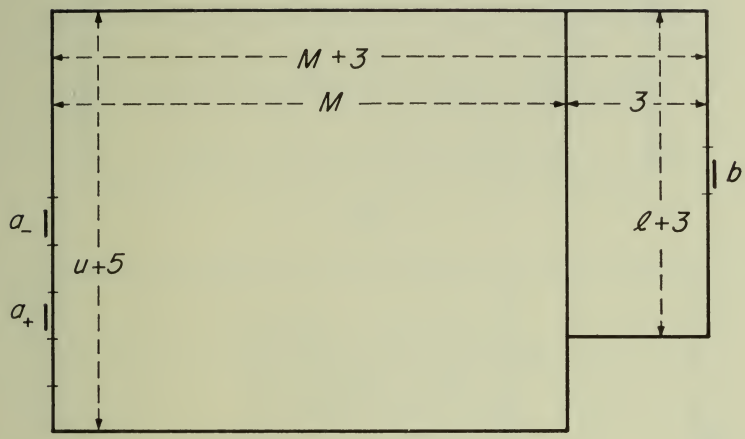


(c)

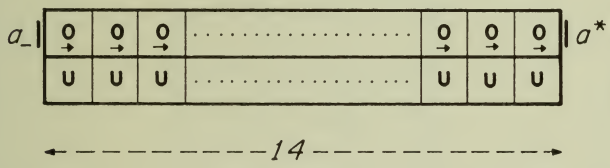


(d)

Fig. 19. Part two

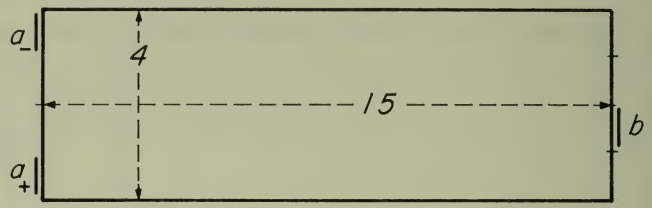


(e)

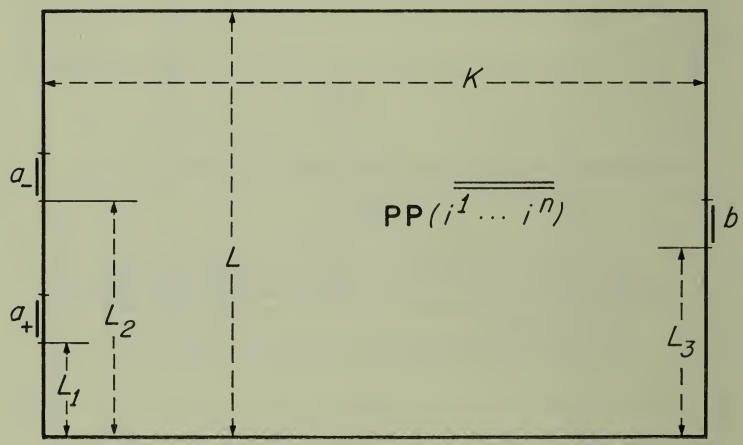


(f)

Fig. 19. Part three



(g)



(h)

Fig. 19. Part four

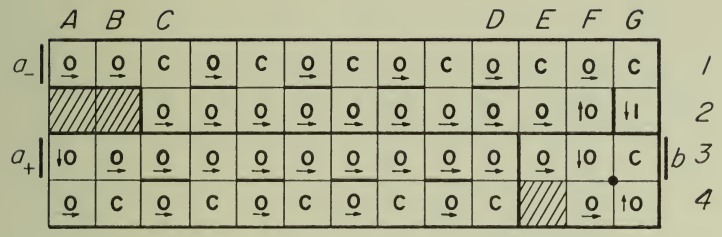


Fig. 20. Alternate periodic pulser $\mathbf{PP}(\overline{1})$

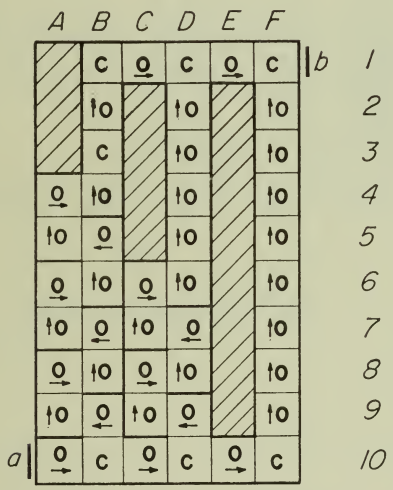


Fig. 21. Decoding organ $\mathbf{D}(\overline{10010001})$

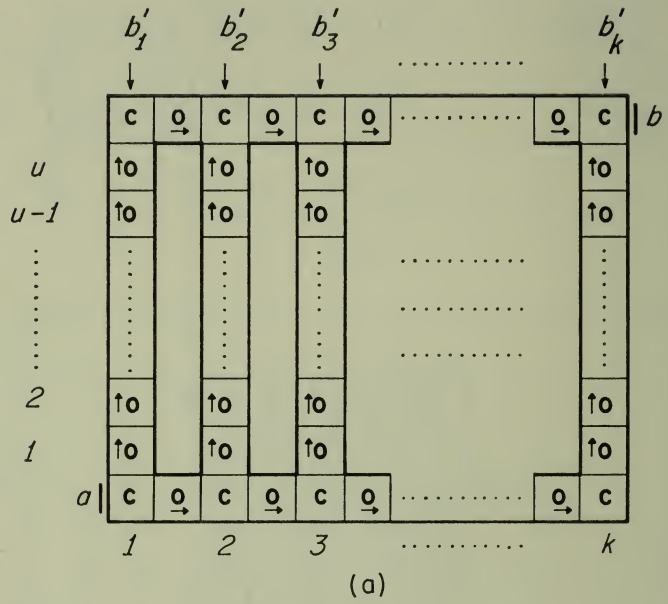
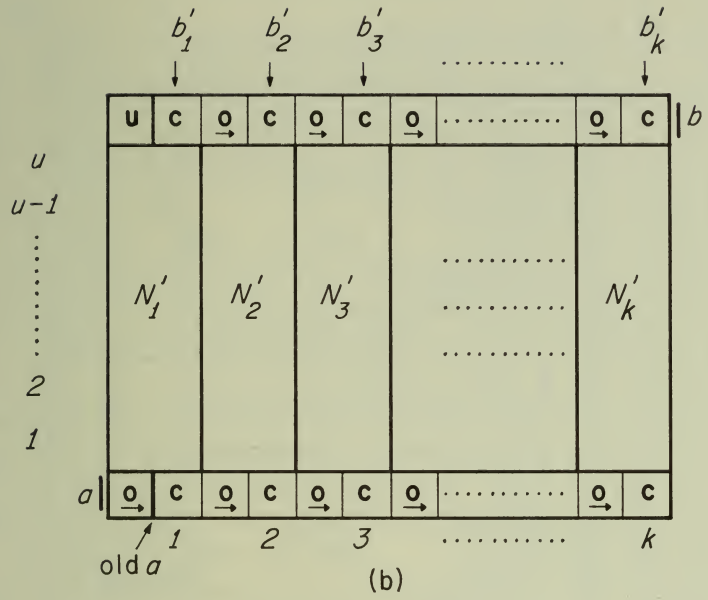
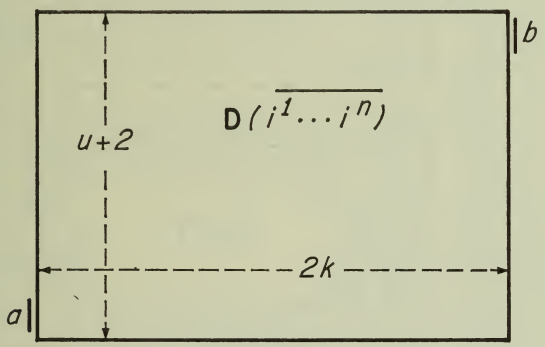


Fig. 22. Design of a decoder



(b)



(c)

Fig. 22. Part two

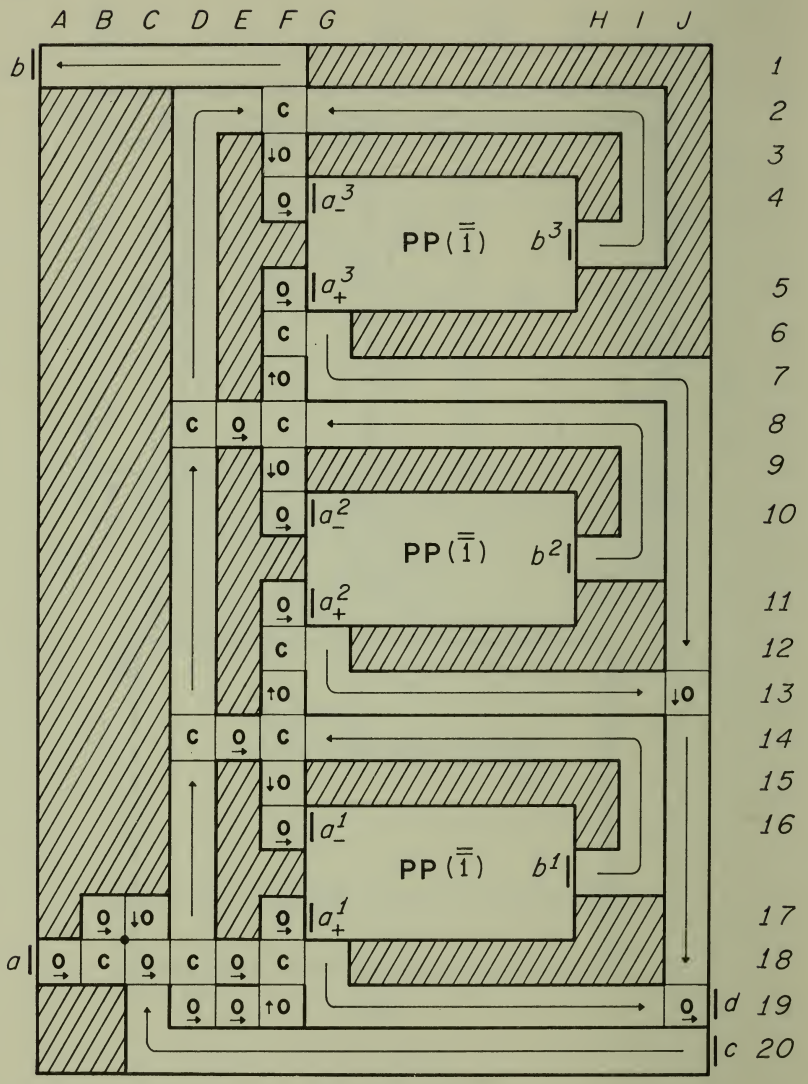


Fig. 23. Triple-return counter Φ

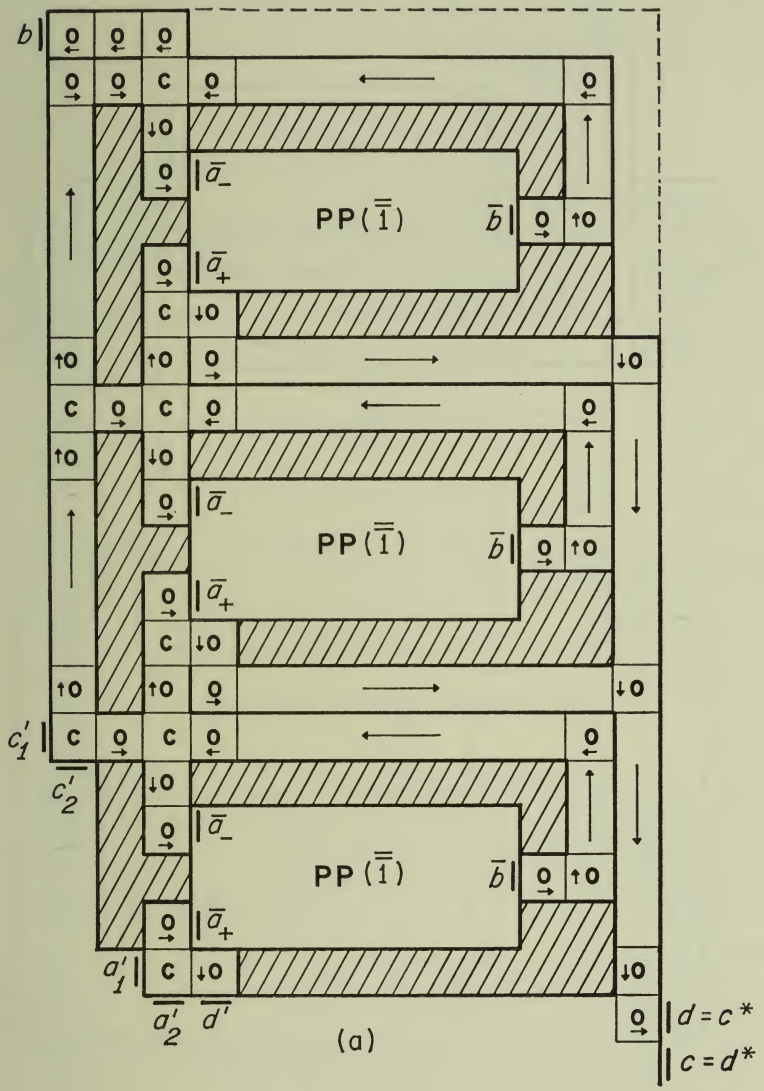
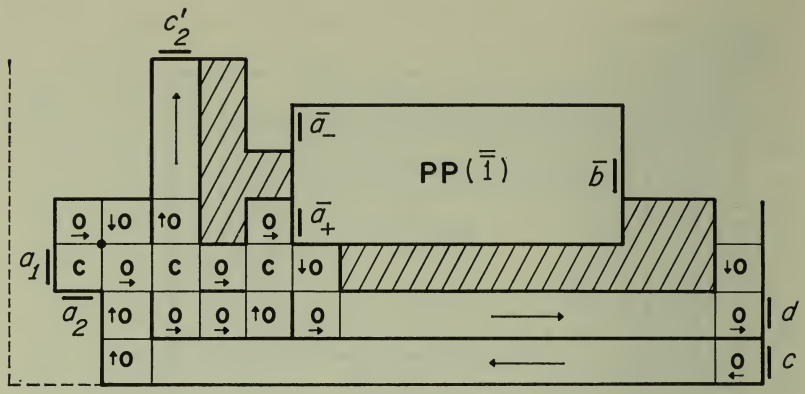
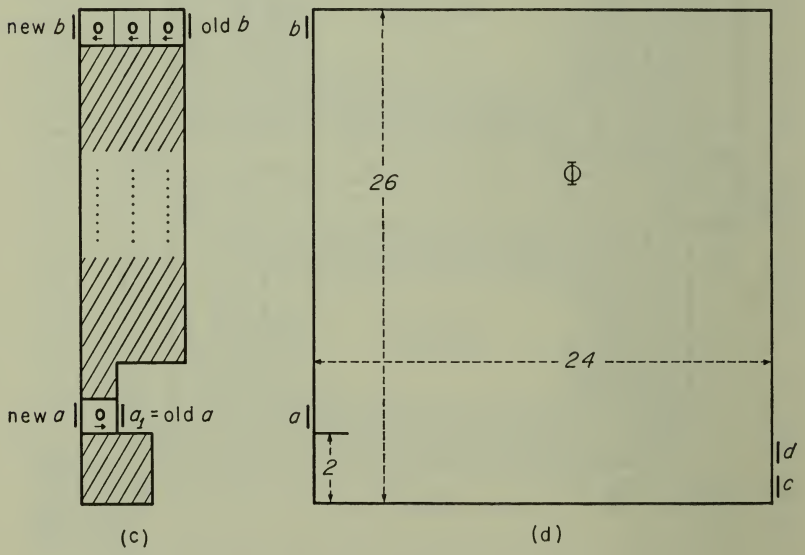


Fig. 24. Design of triple-return counter Φ



(b)



(c) (d)

Fig. 24. Part two

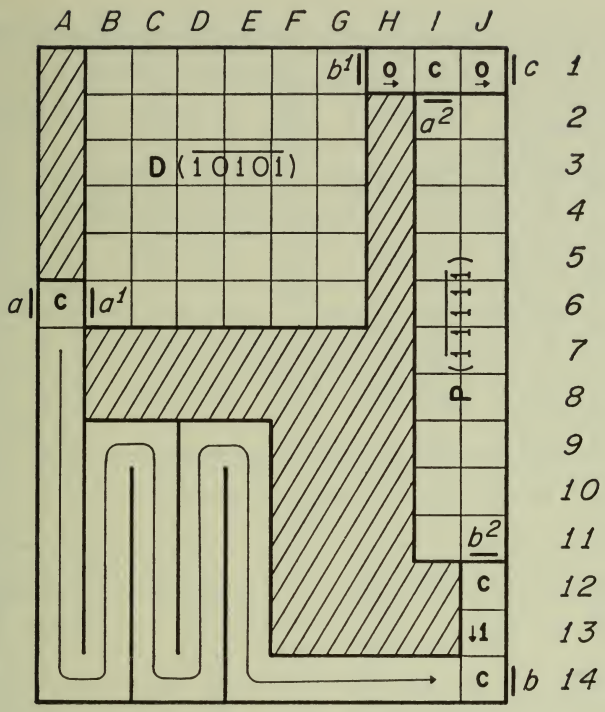


Fig. 25. $\bar{1}$ vs. $\overline{10101}$ discriminator Ψ

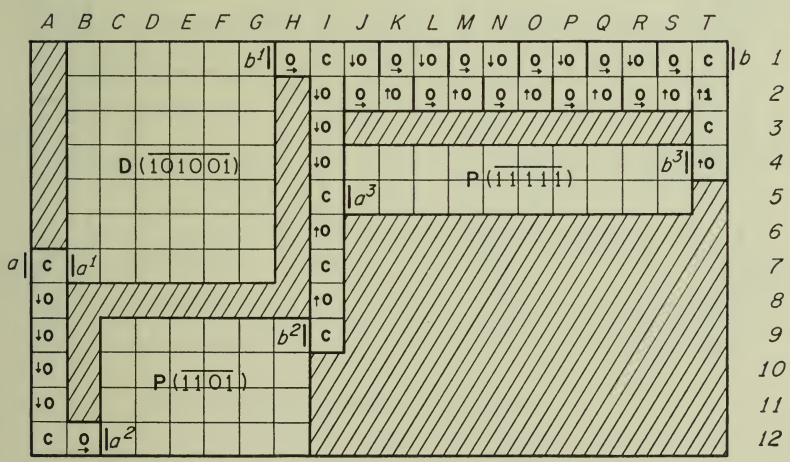


Fig. 26. Recognizer $\mathbf{R}(\overline{101001})$

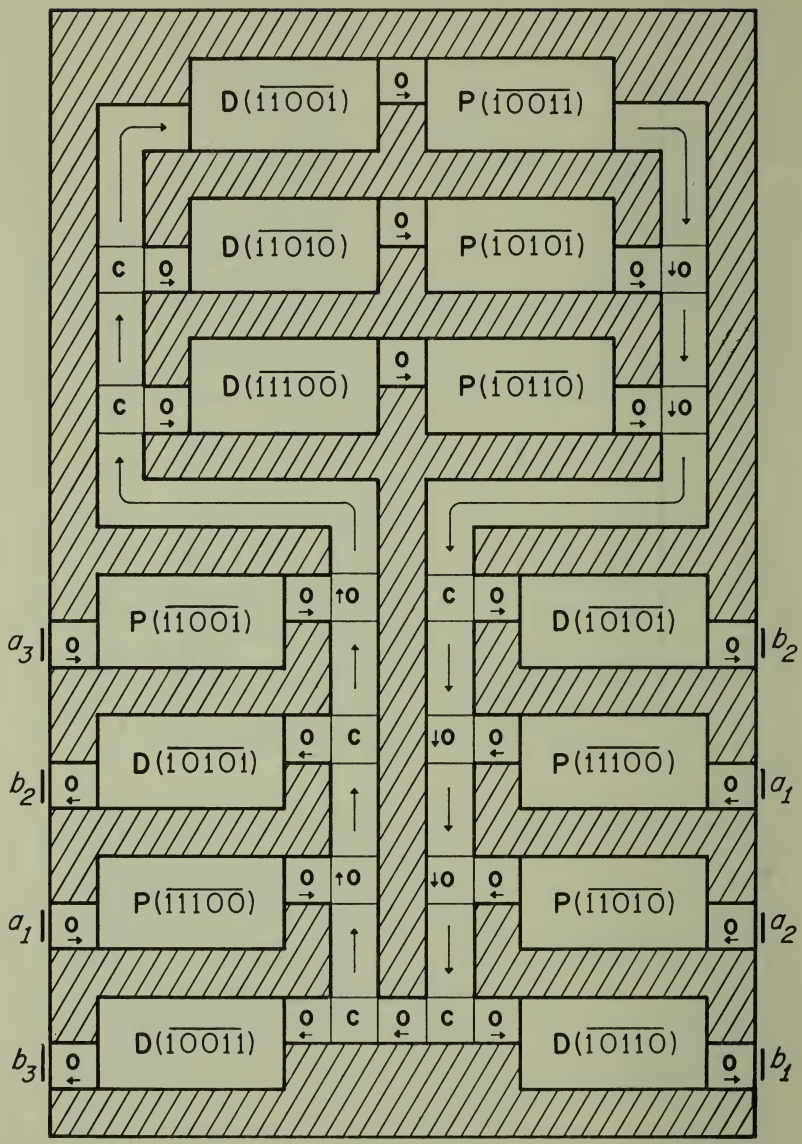
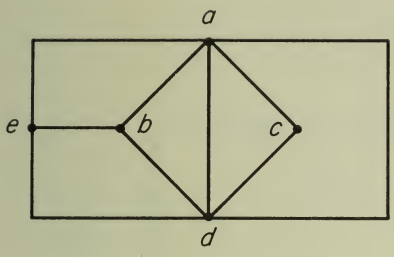
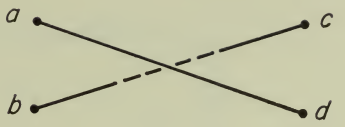


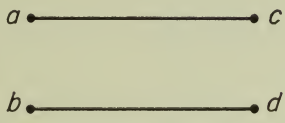
Fig. 27. A coded channel



(a)



(b)



(c)

(d)

(e)

(f)

Fig. 28. Arranging inputs and outputs of a coded channel

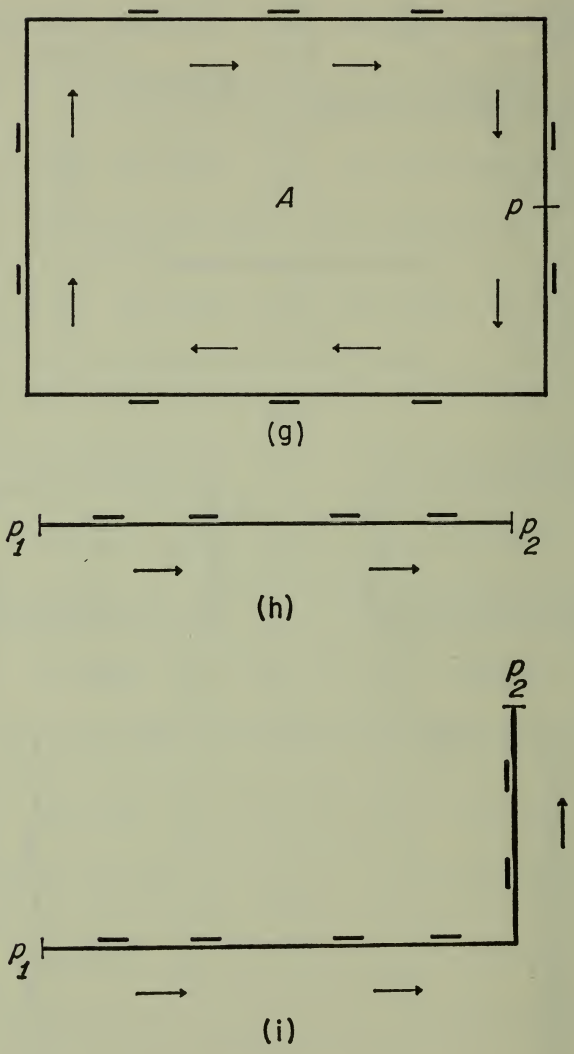
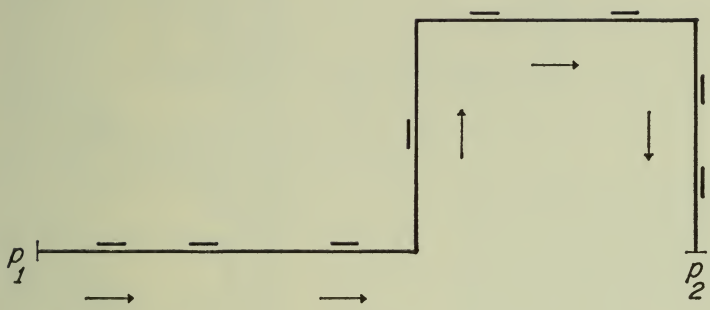
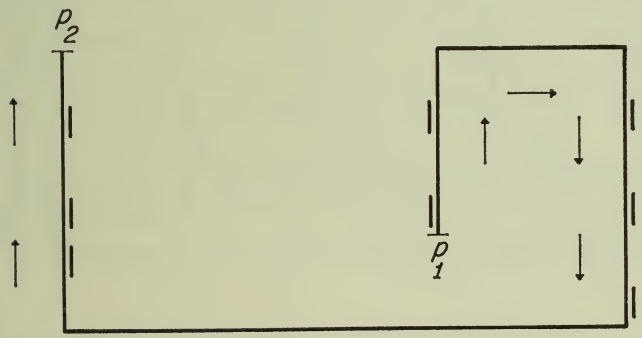


Fig. 28. Part two



(j)



(k)

Fig. 28. Part three

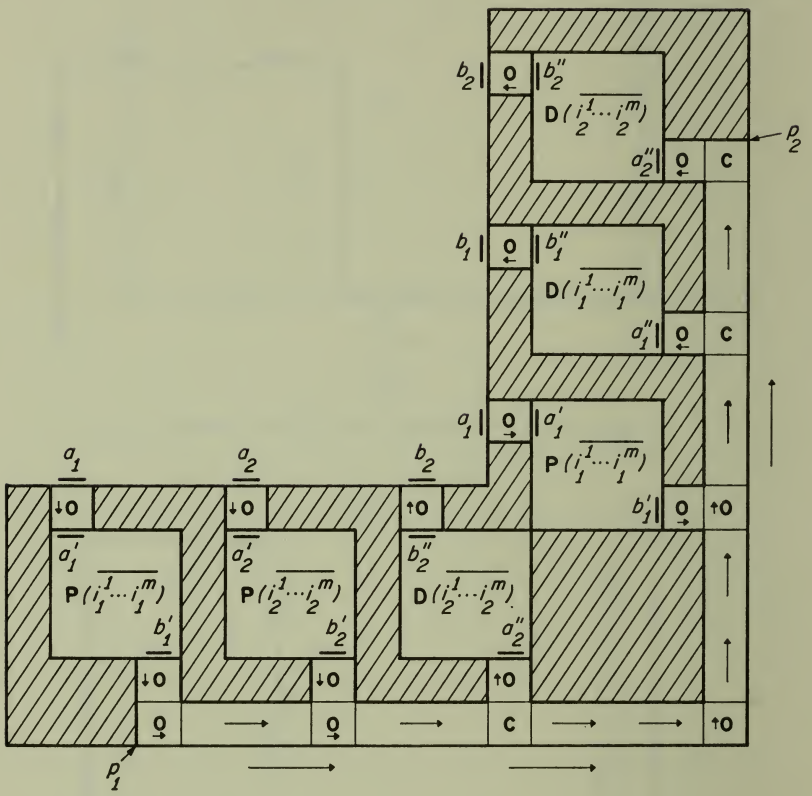


Fig. 29. Pulsers and decoders of a coded channel

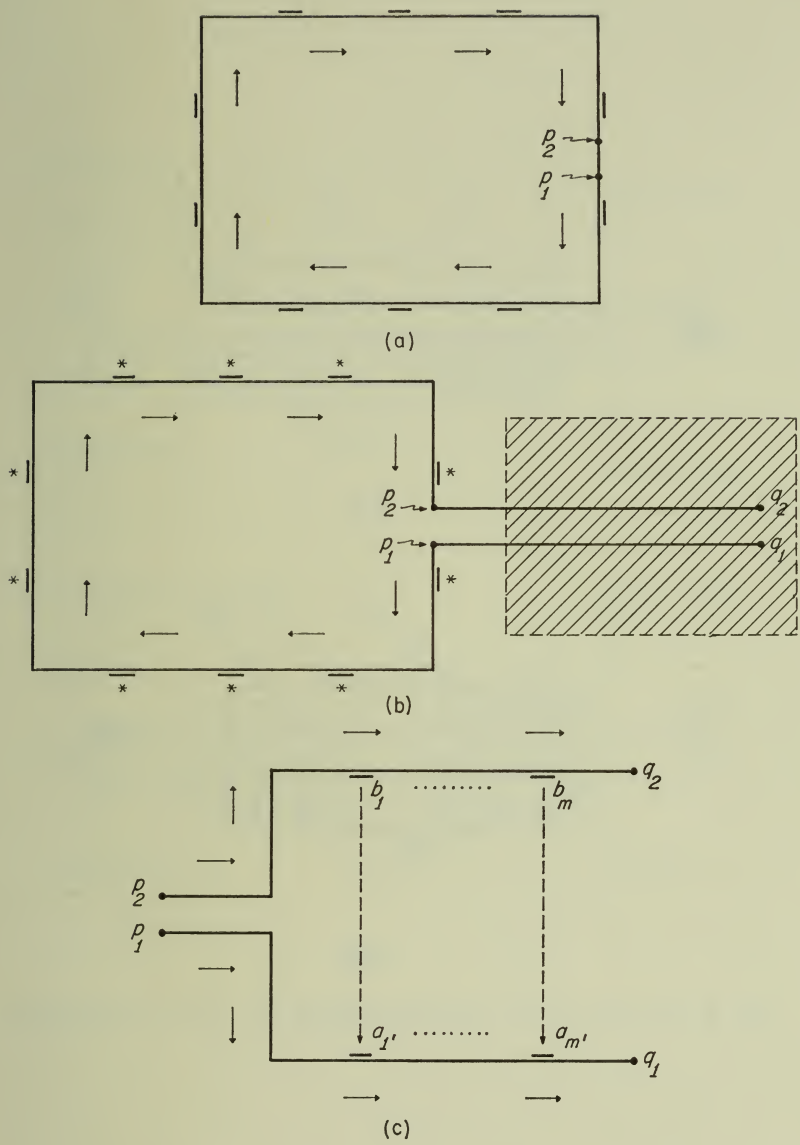


Fig. 30. Cyclicity in a coded channel

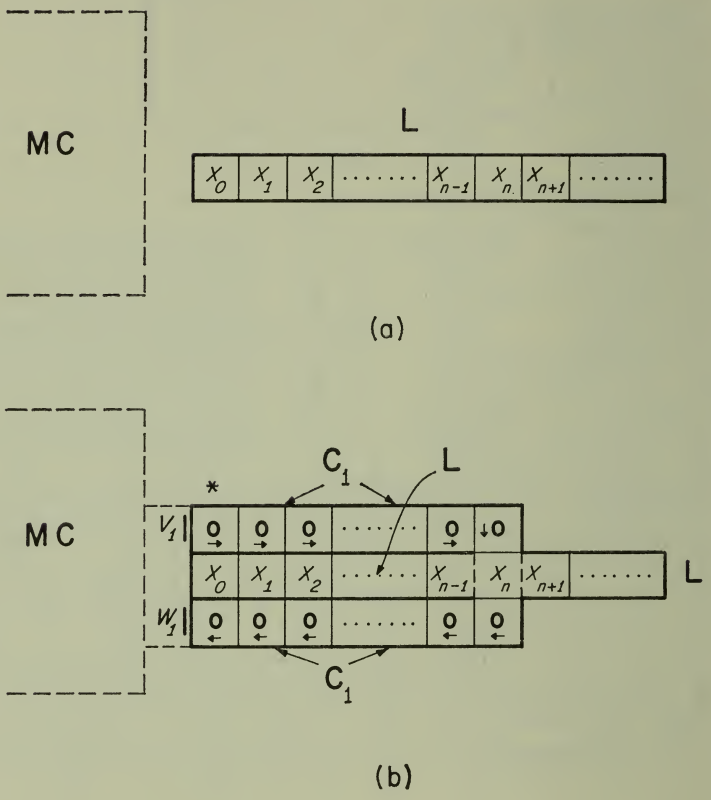
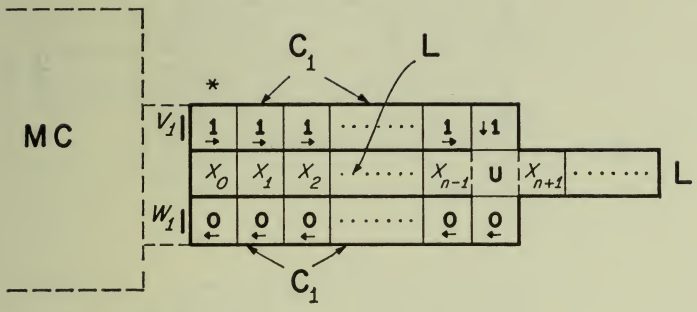
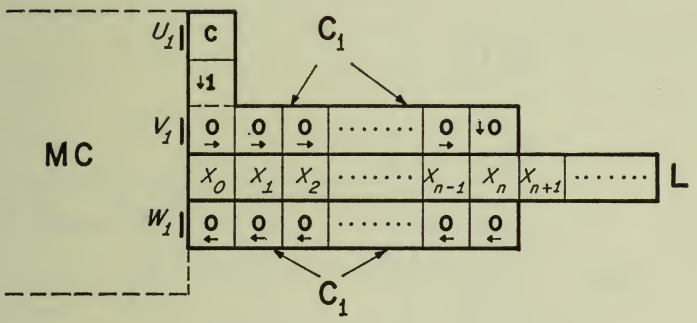


Fig. 31. The linear array **L**, the connecting loop $\mathbf{C}_1$, and the timing loop $\mathbf{C}_2$

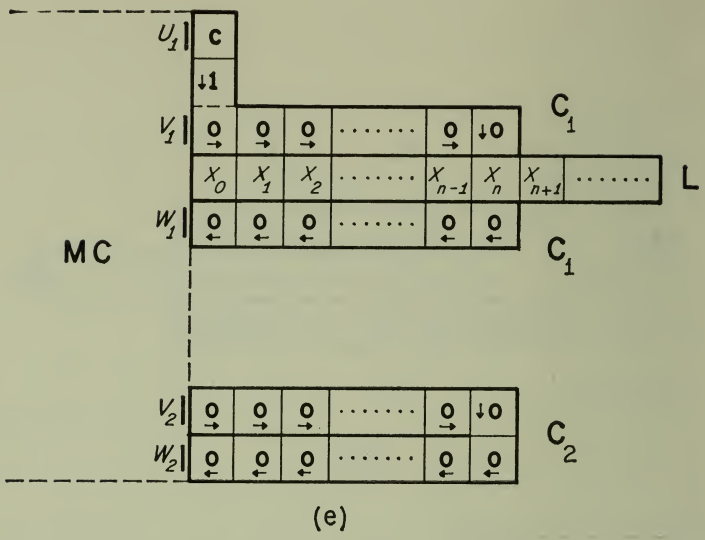


(c)

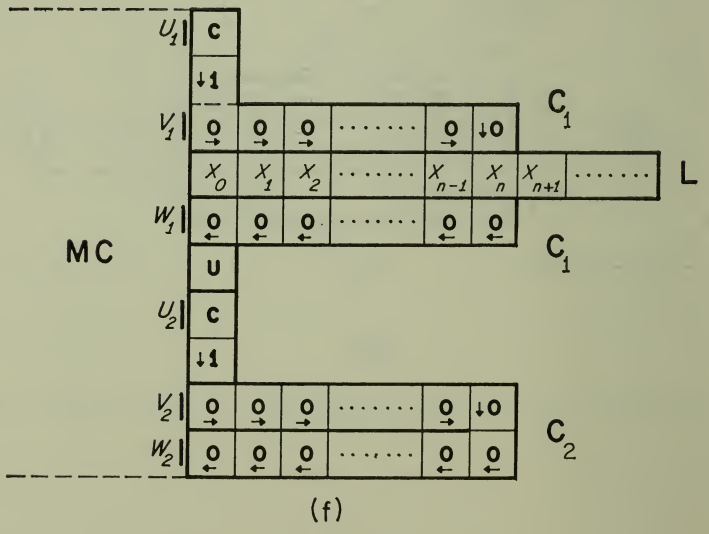


(d)

Fig. 31. Part two

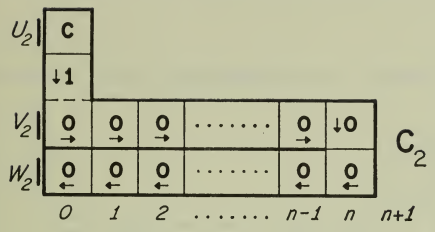


(e)

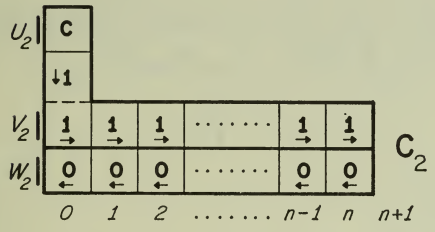


(f)

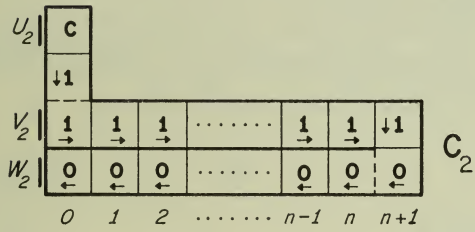
Fig. 31. Part three



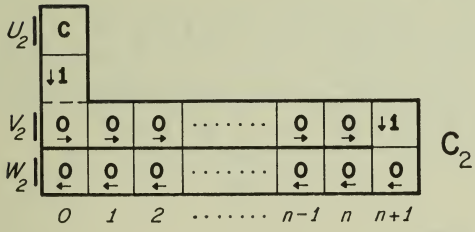
(a)



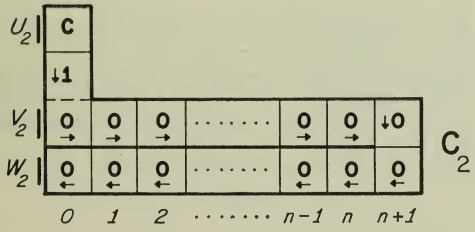
(b)



(c)



(d)



(e)

Fig. 32. Lengthening the timing loop C_2

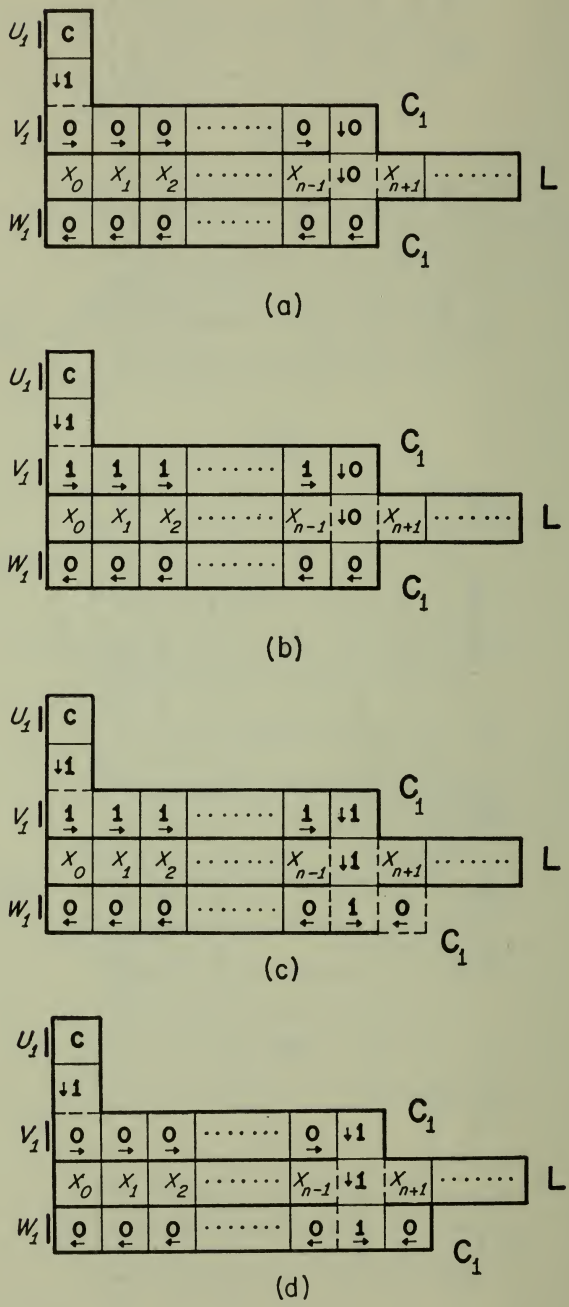
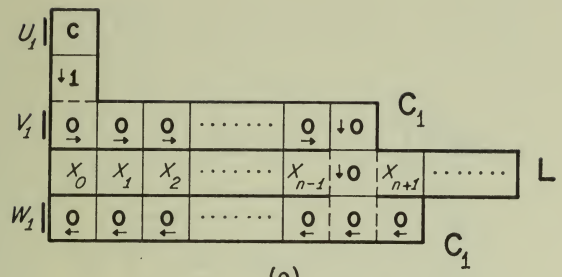
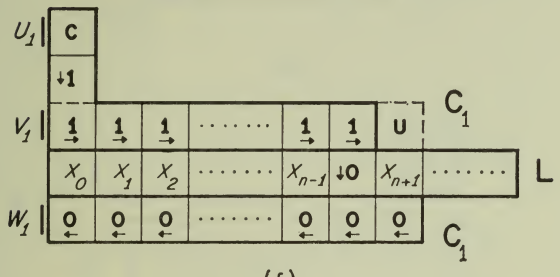
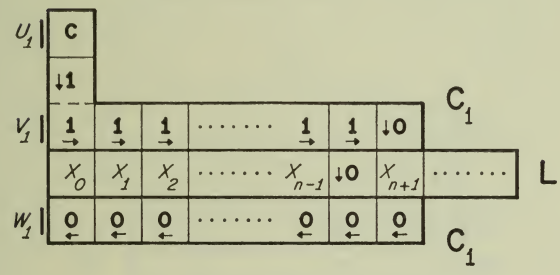


Fig. 33. Lengthening the connecting loop C_1







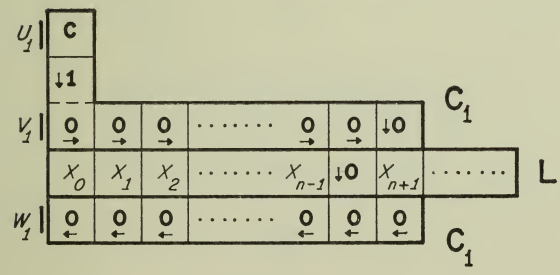


Fig. 33. Part two

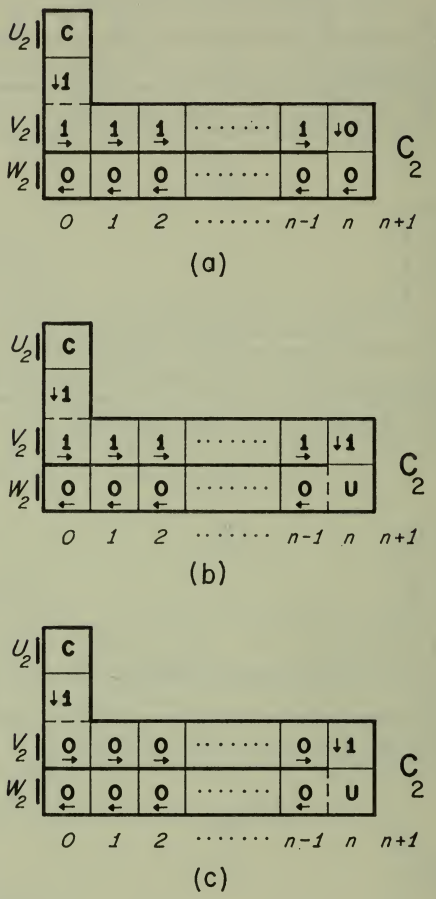
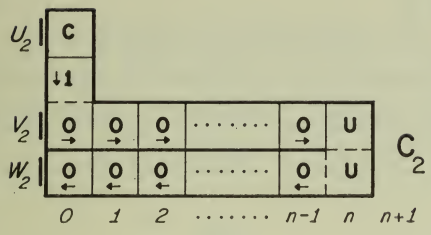
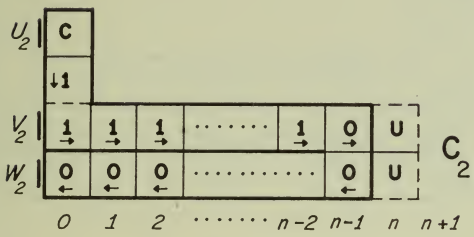


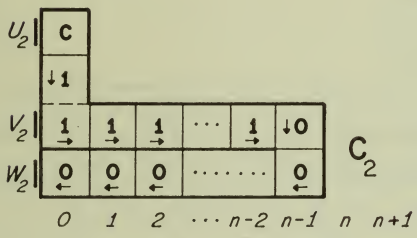
Fig. 34. Shortening the timing loop C_2



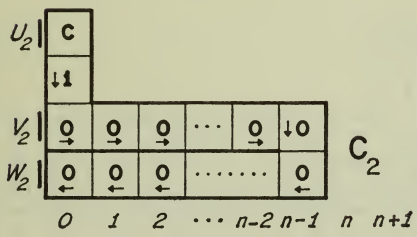
(d)



(e)

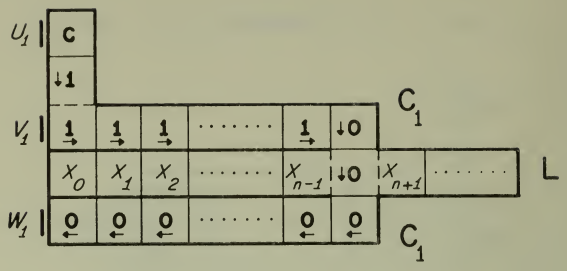


(f)

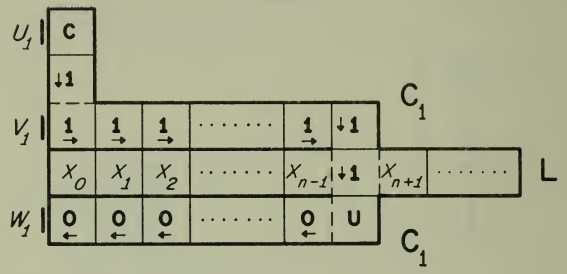


(g)

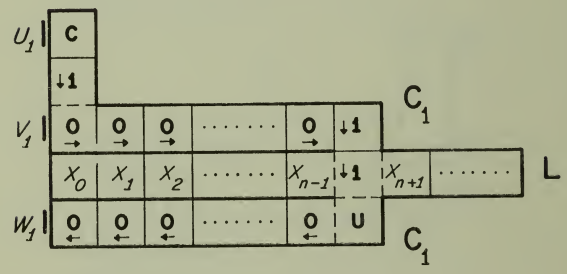
Fig. 34. Part two



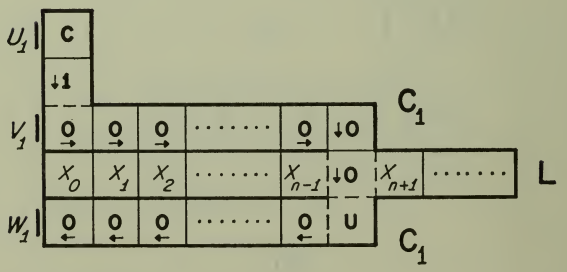
(a)



(b)

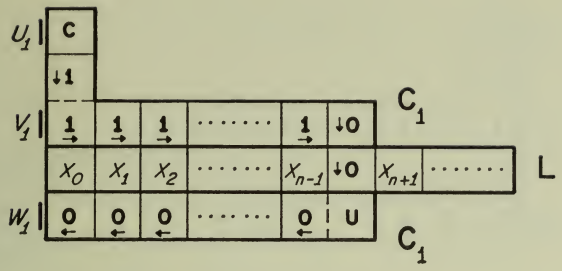


(c)

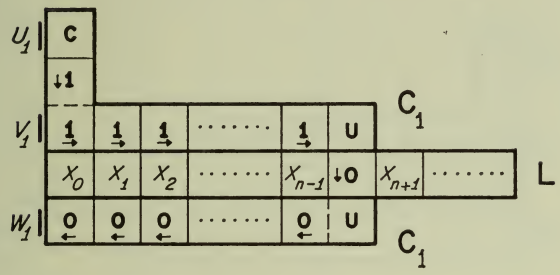


(d)

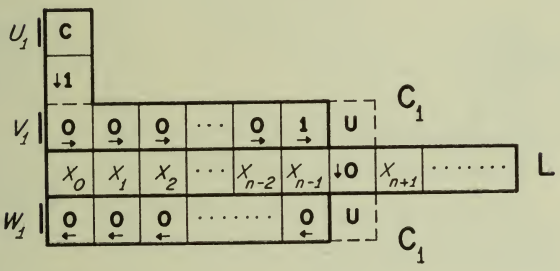
Fig. 35. Shortening the connecting loop C_1



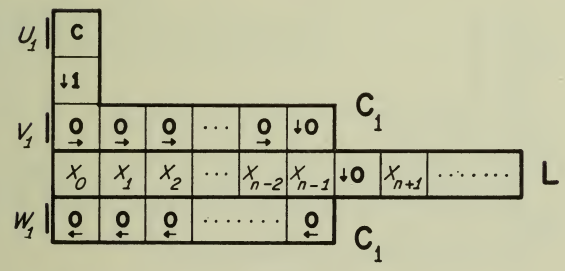
(e)



(f)

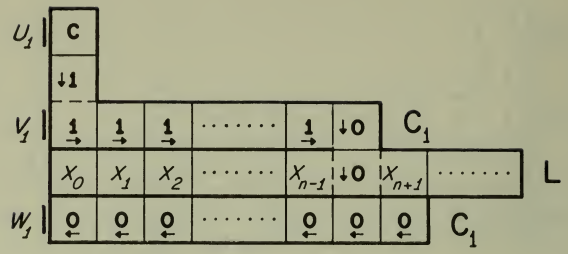


(g)

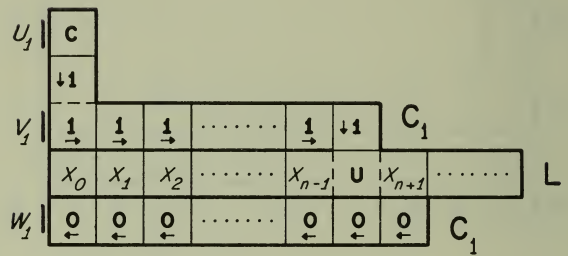


(h)

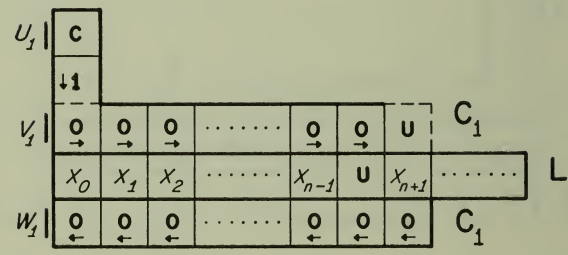
Fig. 35. Part two



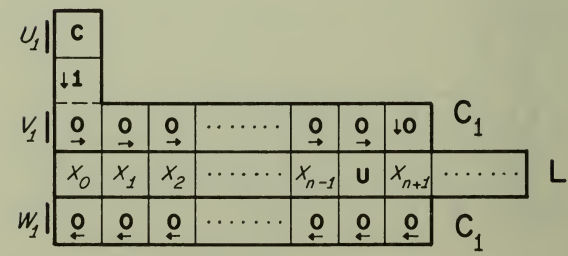
(a)



(b)



(c)



(d)

Fig. 36. Writing "zero" in cell x_n of the linear array **L** when lengthening $\mathbf{C}_1$

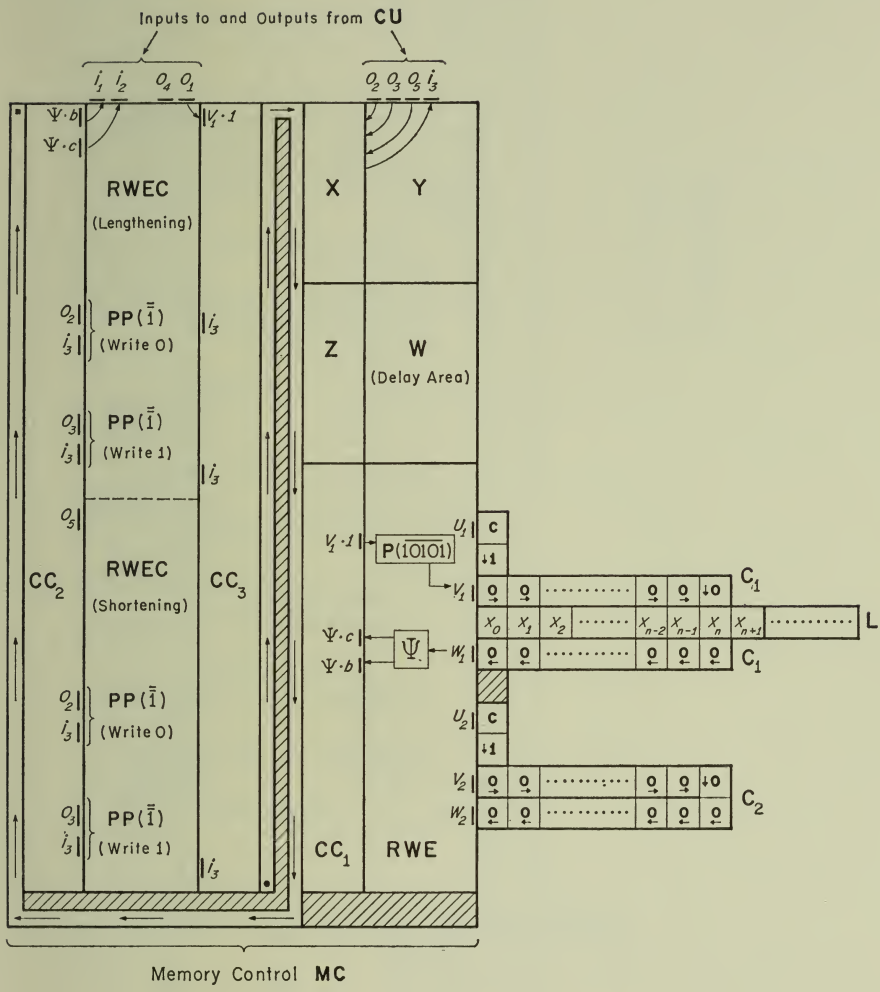


Fig. 37. Tape unit with unlimited memory capacity.
Note: the various organs and units are not drawn to scale.

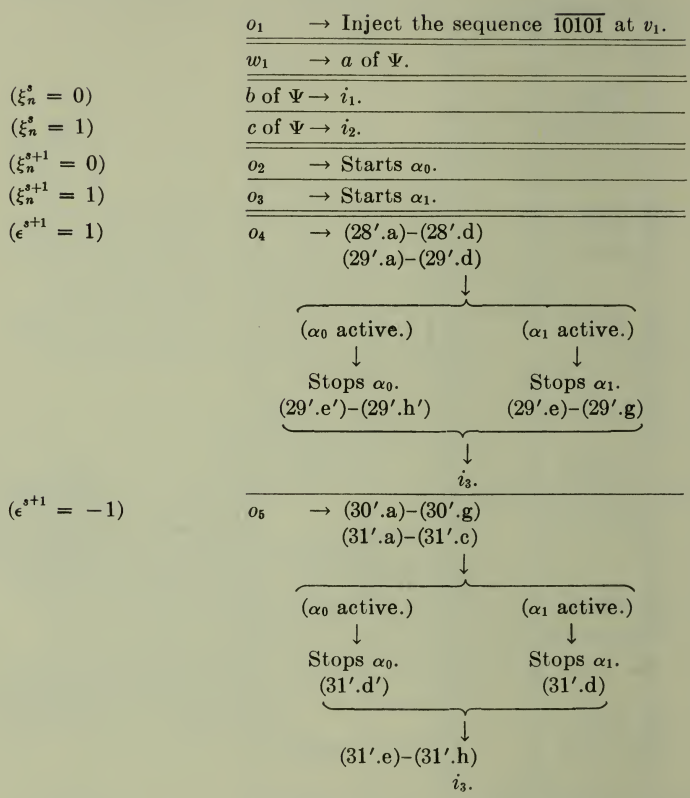


Fig. 38. The logical structure of the procedure followed by the memory control **MC**.

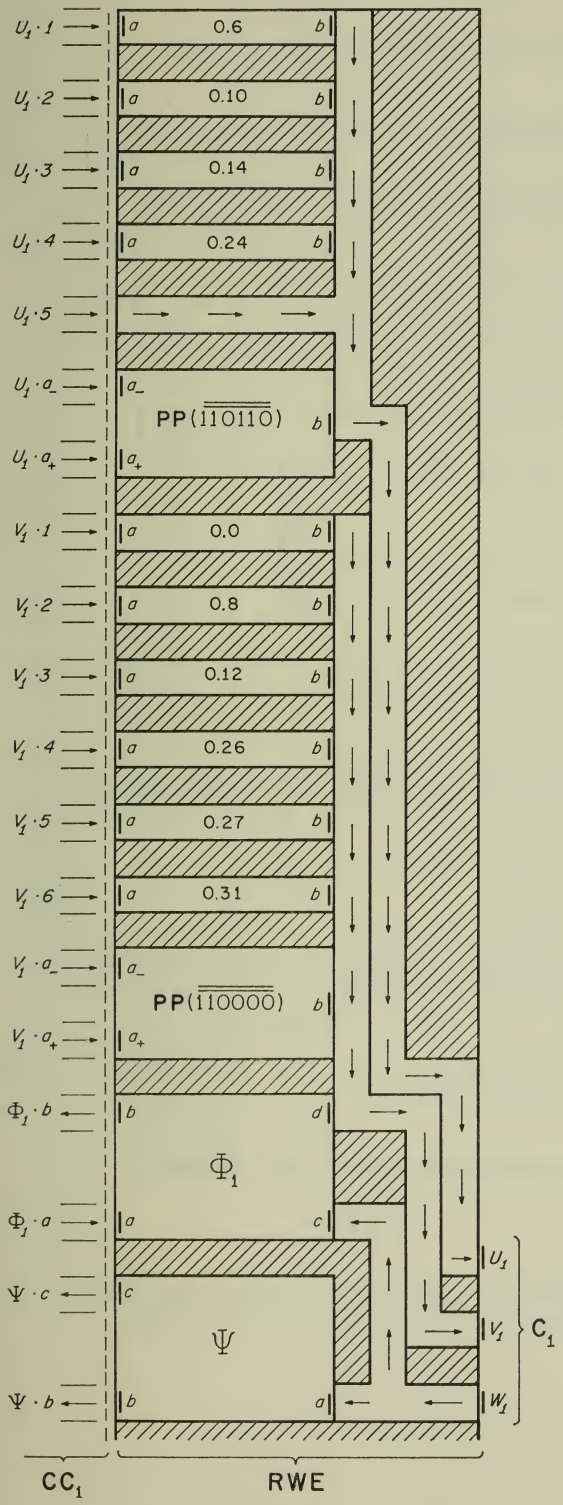


Fig. 39. Read-write-erase unit **RWE**. (a) Upper part of **RWE**

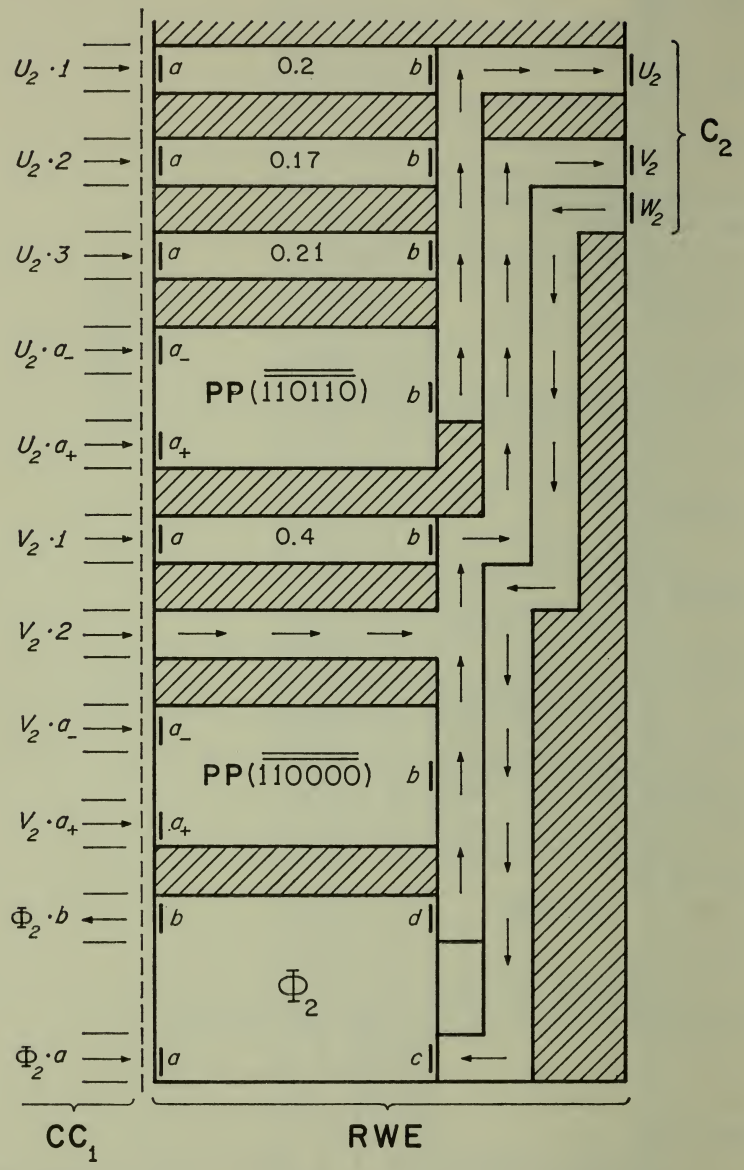


Fig. 39. Read-write-erase unit **RWE**. (b) Lower part of **RWE**. Note: parts (a) and (b) of Figure 39 constitute one continuous drawing.

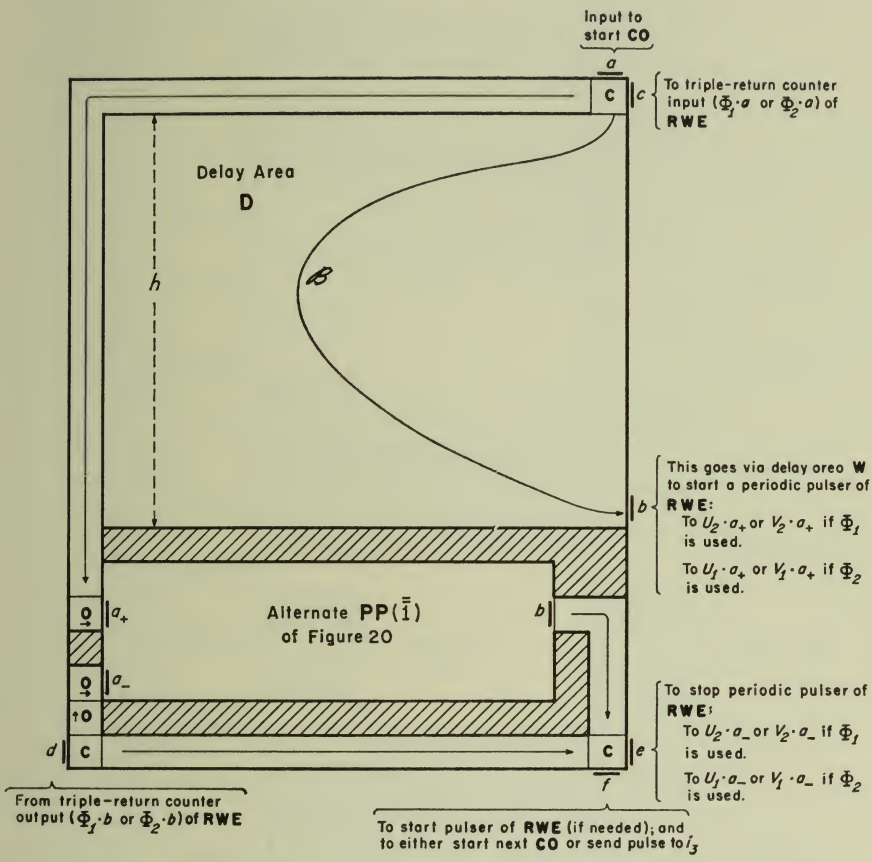


Fig. 40. Control organ **CO**

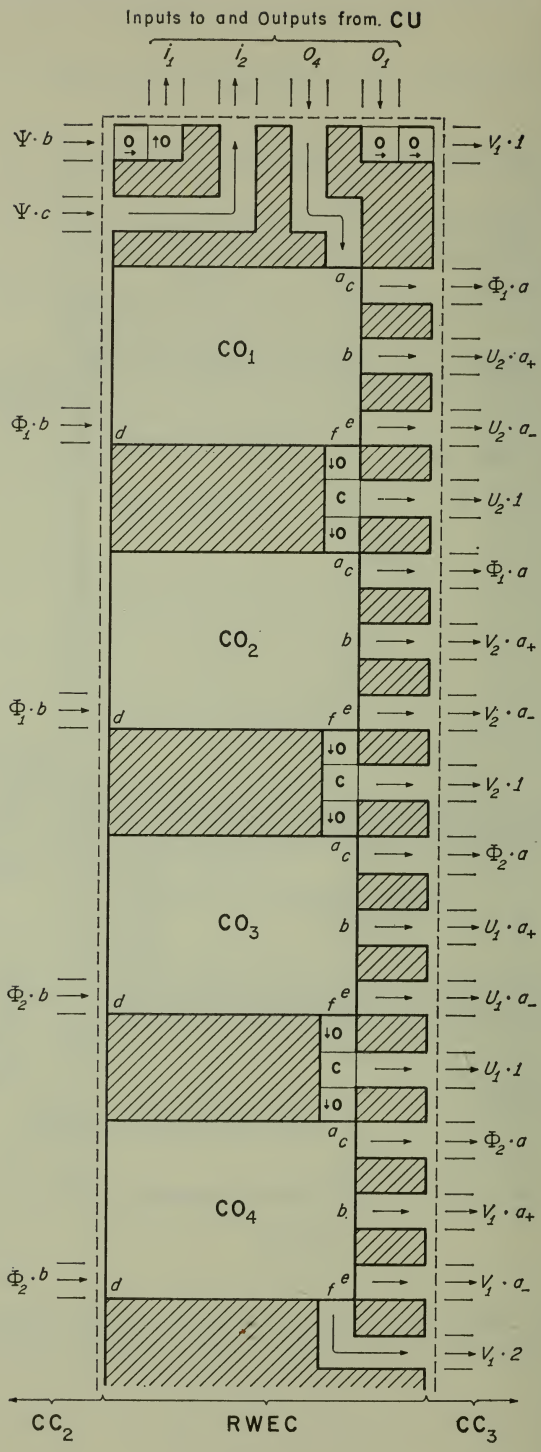


Fig. 41. Read-write-erase control **RWEC**. (a) Control organs for lengthening C_2 (CO_1 and CO_2) and for lengthening the lower part of C_1 (CO_3 and CO_4).

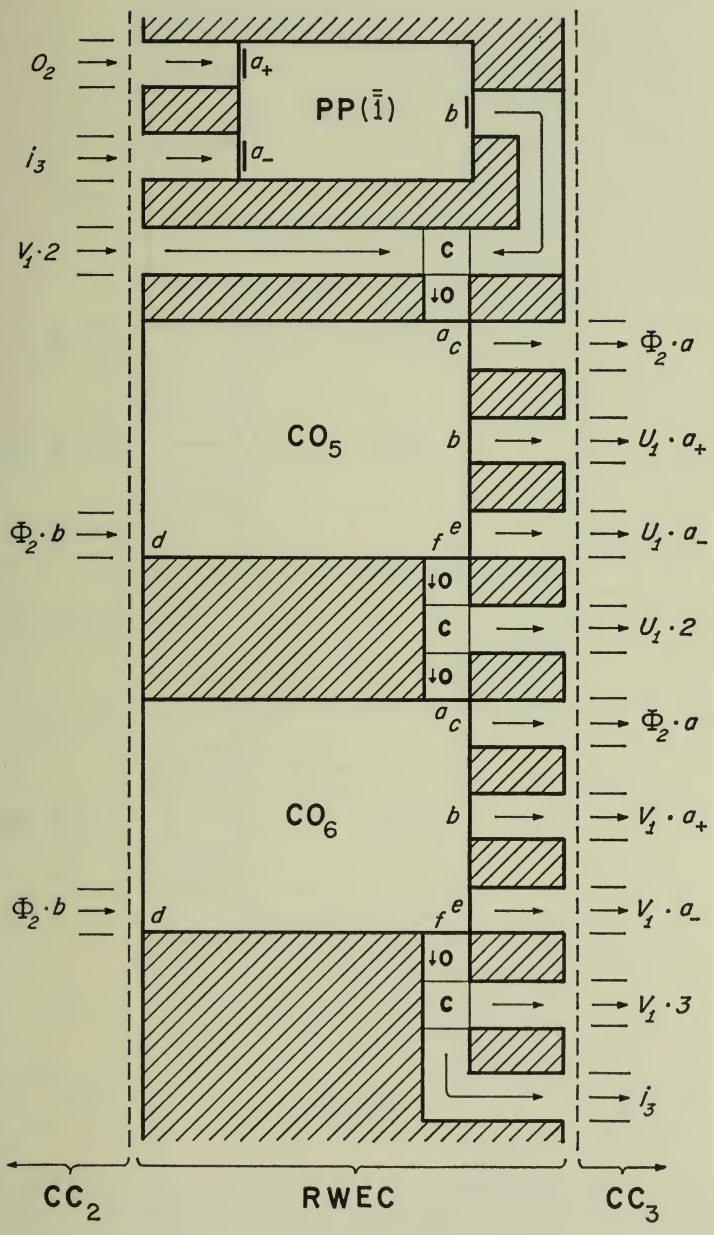


Fig. 41. Read-write-erase control **RWEC**. (b) **PP**($\bar{1}$) to store the fact that a "zero" is to be written in z_n, and control organs for writing a zero and lengthening the upper part of $\mathbf{C}_1$. (This figure is continued from p. 358.)

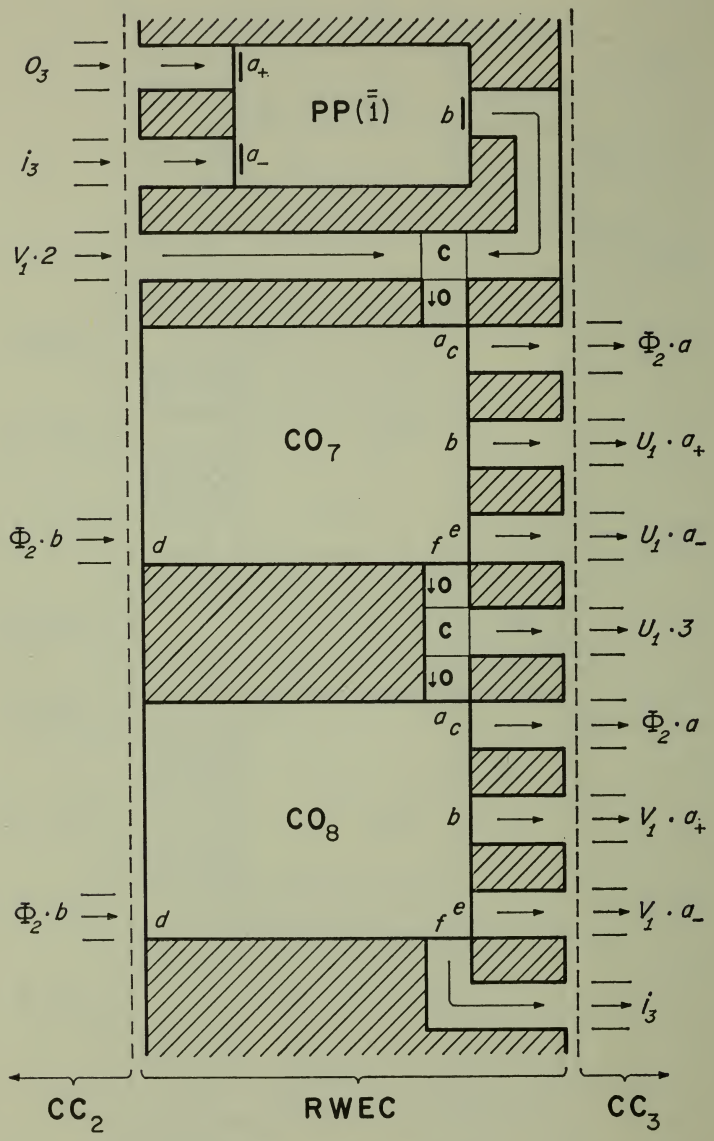


Fig. 41. Read-write-erase control **RWEC**. (c) **PP**($\bar{1}$) to store the fact that a "one" is to be written in x_n, and control organs for leaving a one in cell x_n and lengthening the upper part of C_1. (This figure is continued from p. 359.)

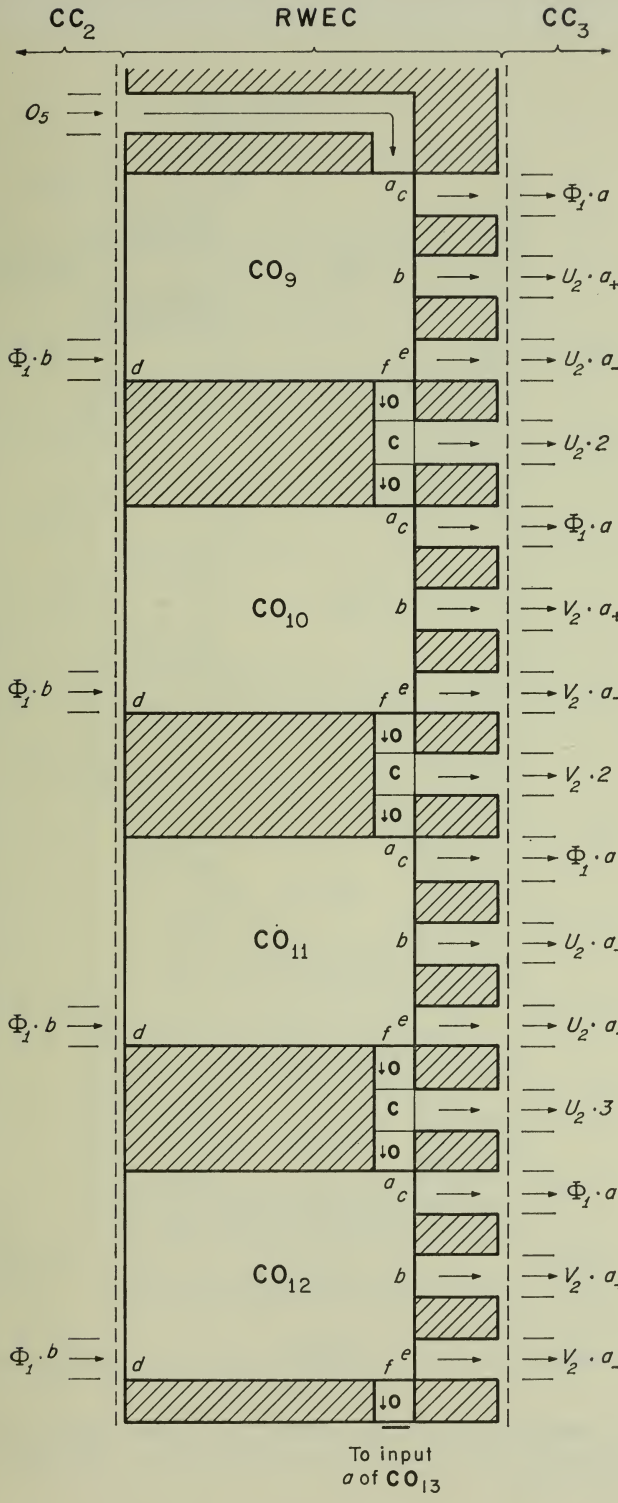


Fig. 41. Read-write-erase control **RWEC**. (d) Control organs for shortening C_2 . (This figure is continued from p. 360.)

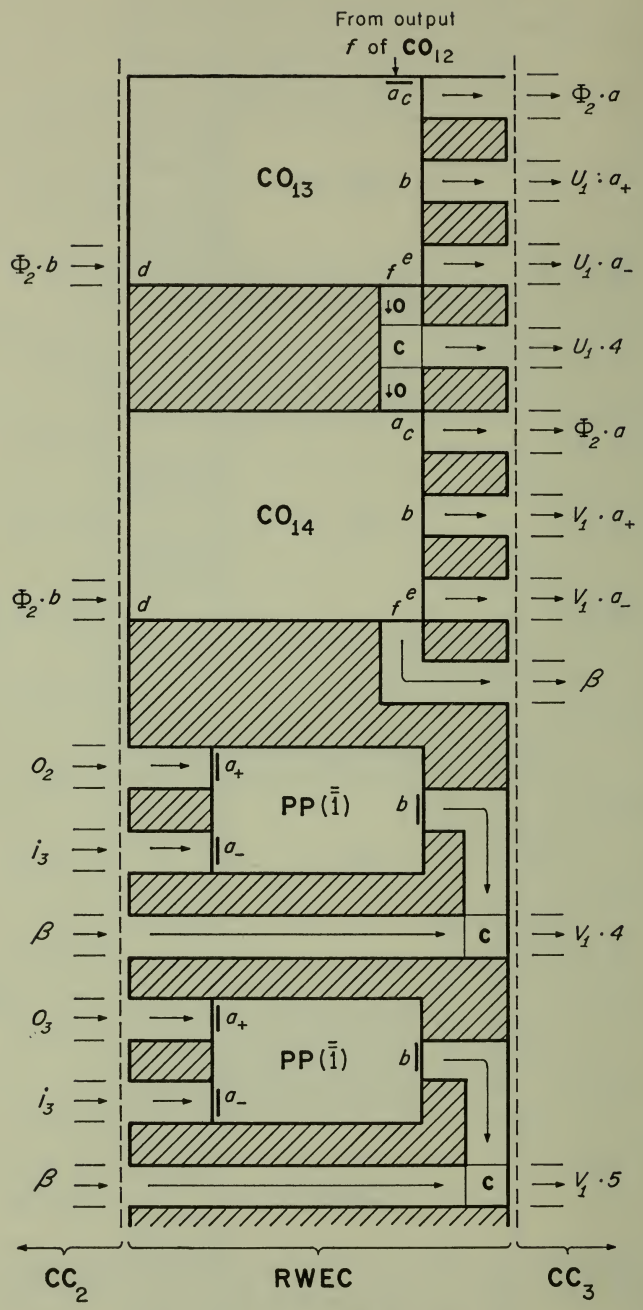


Fig. 41. Read-write-erase control **RWEC**. (e) Control organs and **PP**($\bar{1}$) for shortening the lower part of C_1 and writing in cell x_n . (This figure is continued from p. 361.)

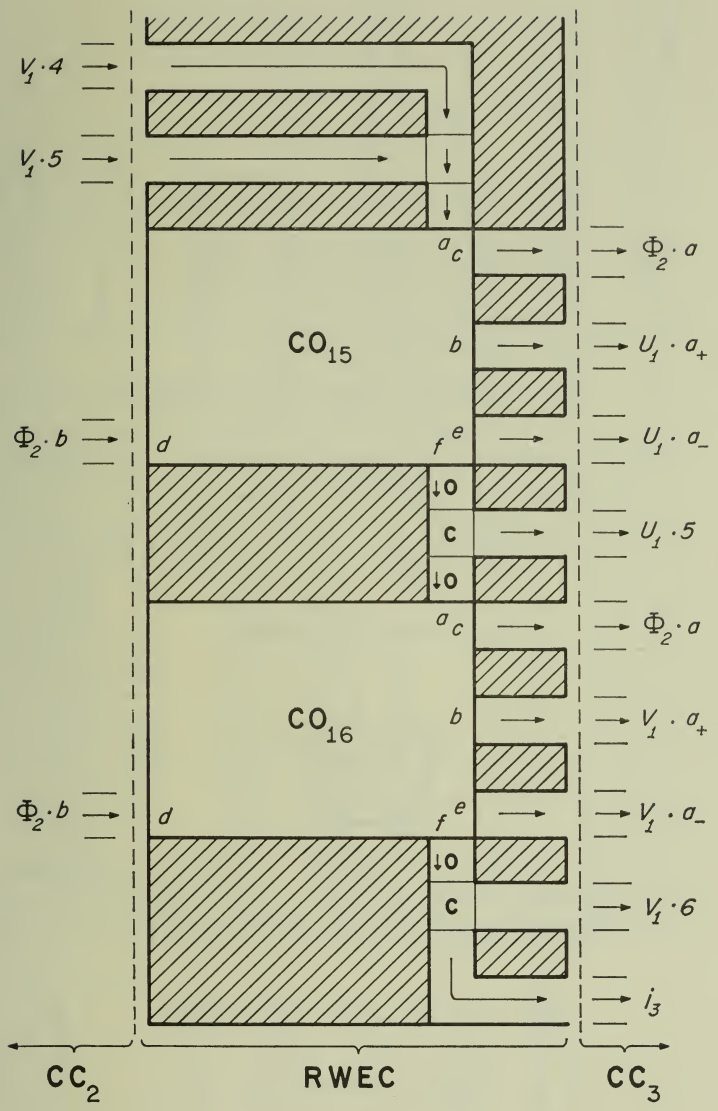
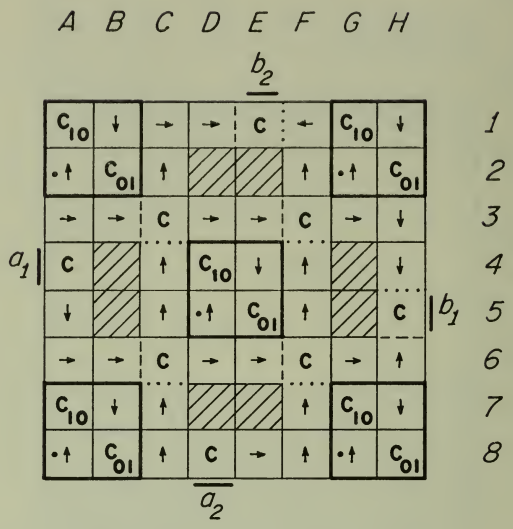
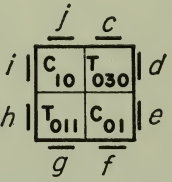


Fig. 41. Read-write-erase control **RWEC**. (f) Control organs for shortening the upper part of C_1. (This completes Figure 41.)



(a) Crossing organ



Output from i and j: 101010
Output from e and f: 010101

(b) Initial state of clock

Fig. 42. Crossing organ

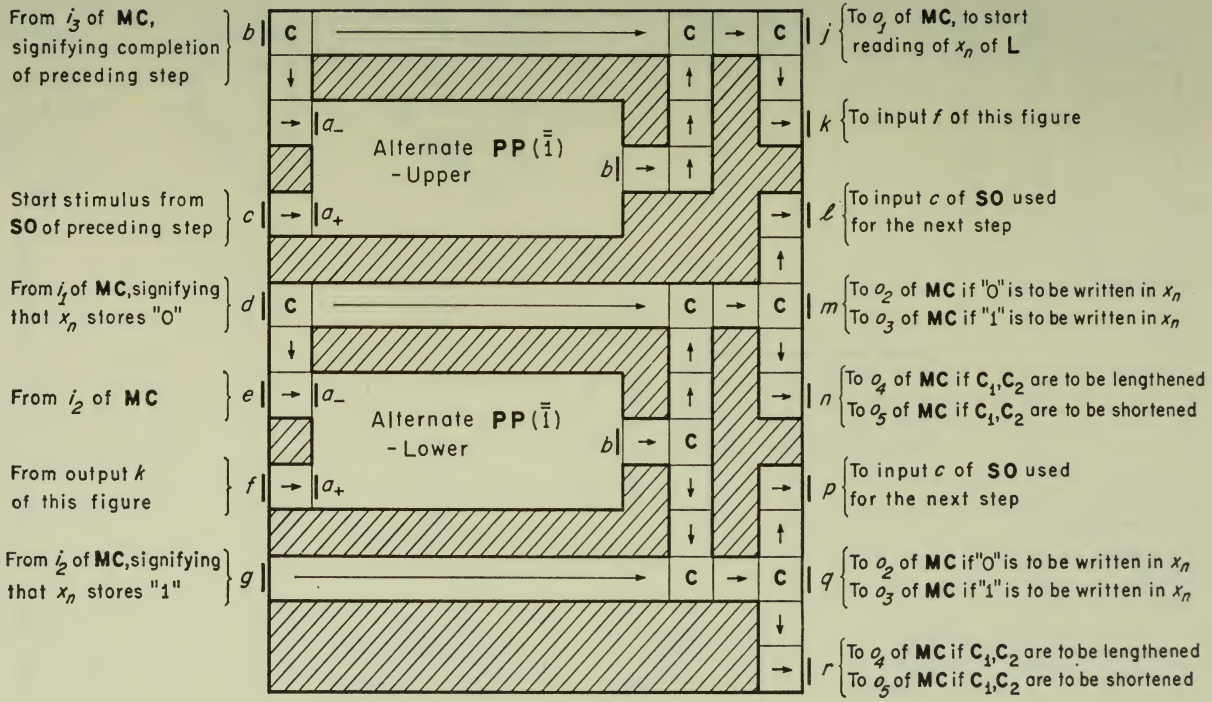
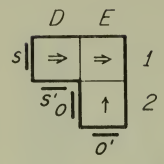
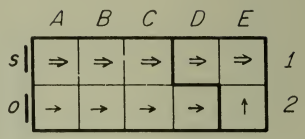


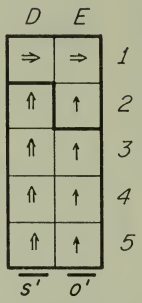
Fig. 43. State organ **SO** of a finite automaton **FA**



(a) Head of constructing arm



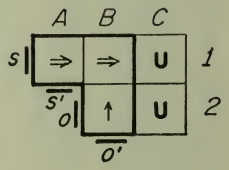
(b) Head fed from left



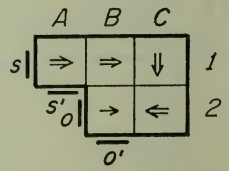
(c) Head fed from below

Fig. 44. Constructing arm

(a)
Starting with



(b)
⇓ ⇐ U → into s or s' produces



(c)
U ↑ U ⇒ into o or o' produces

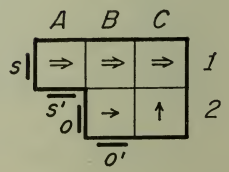
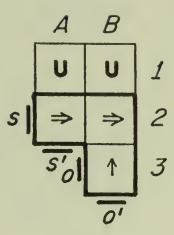


Fig. 45. Horizontal advance of constructing arm

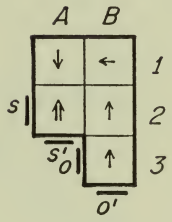
(a)

Starting with



(b)

U ↑ ← ↓ ⇑ into *o* or *o'* produces



(c)

U ⇒ U ⇒ into *s* or *s'* produces

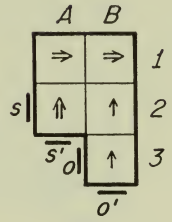
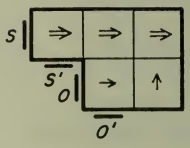
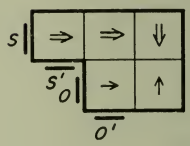


Fig. 46. Vertical advance of constructing arm

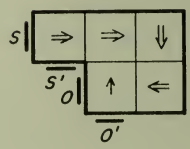
(a)
Starting with



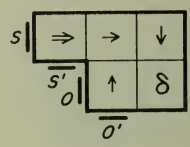
(b)
U ⇓ into o or o' produces



(c)
U ⇐ U ↑ into s or s' produces



(d)
U → U ↓ U δ into o or o' produces



(e)
U ⇒ U γ into s or s' produces

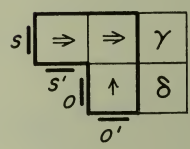
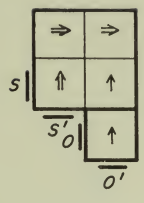
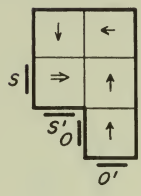


Fig. 47. Horizontal retreat of constructing arm with construction of γ and δ

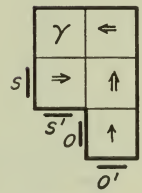
(a)
Starting with



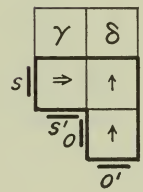
(b)
U ← U ↓ U⇒ into o or o' produces



(c)
U ⇑ U ⇐ U γ into s or s' produces



(d)
U ↑ δ into o or o' produces



(e)
U ⇒ into s or s' produces

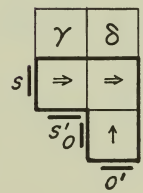
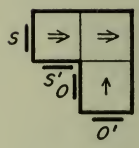
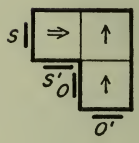


Fig. 48. Vertical retreat of constructing arm with construction of γ and δ

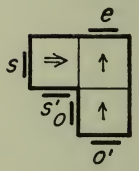
(a)
Starting with



(b)
U ↑ into o or o' produces



(c)
A pulse into o or o' exits at e as the stimulus to start the secondary automaton



(d)
U ⇒ into s or s' produces

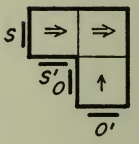


Fig. 49. Injection of starting stimulus into the secondary automaton

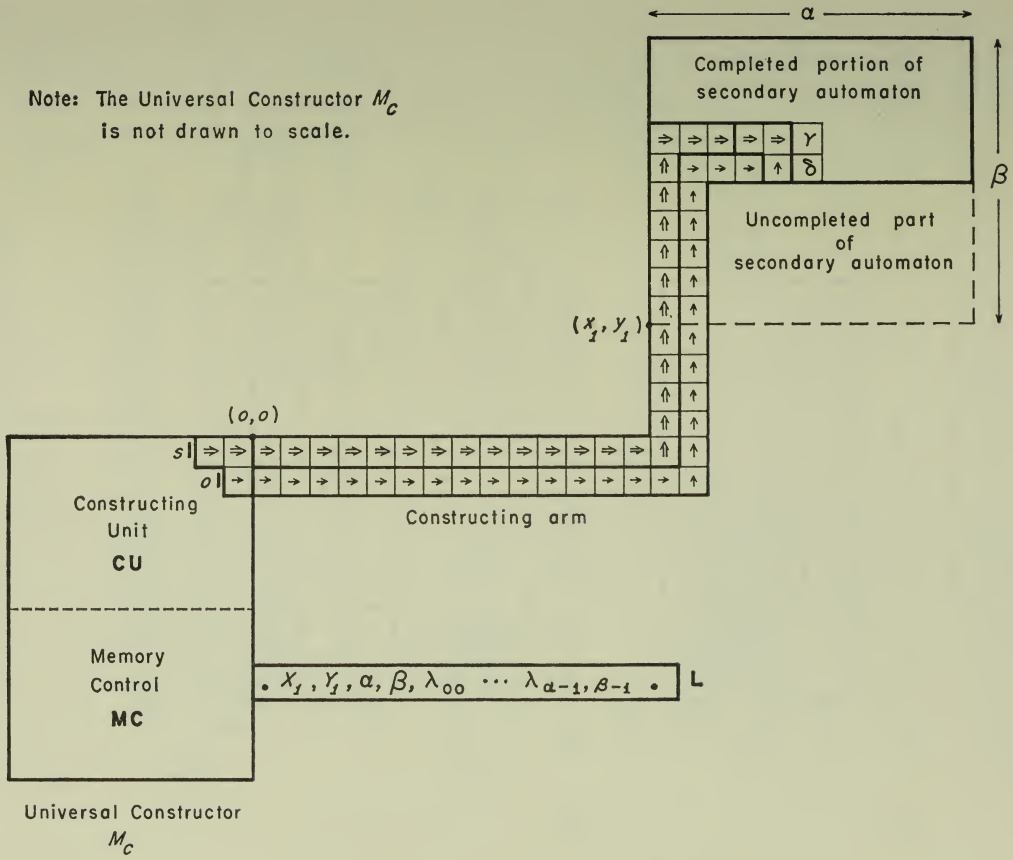
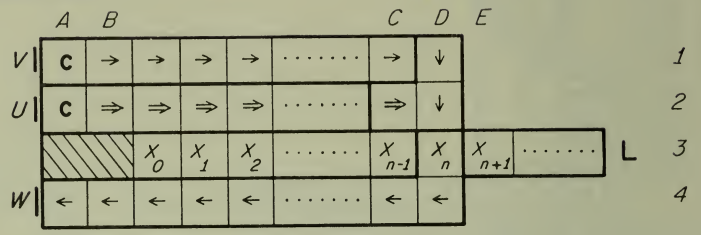


Fig. 50. Operation of the constructing arm

(a) Start with



in which the reading loop passes through cell X_n.

(b) The sequence $\overline{1}\,0\,1\,0\,1$ into V produces:

(1) $\overline{1}$ at W if X_n is U (representing "zero")

(2) $\overline{1\,0\,1\,0\,1}$ at W if X_n is $\downarrow$ (representing "one")

(3) and leaves cell X_n in state $\downarrow$:

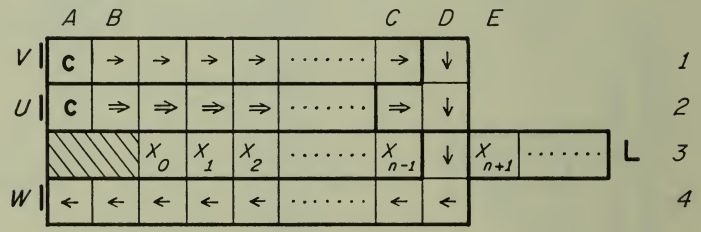


Fig. 51. New method of operating the linear array **L**

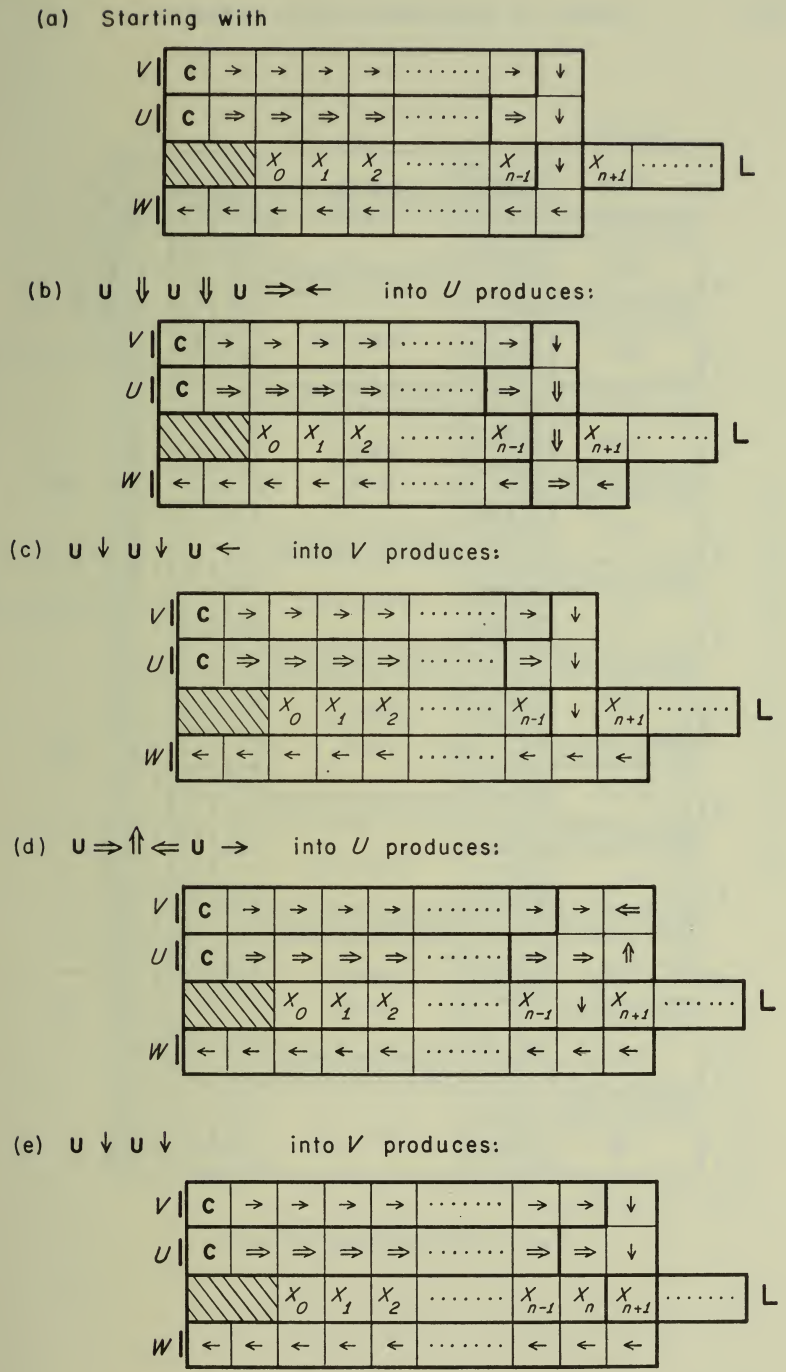


Fig. 52. Writing "one" in cell x_n and lengthening the reading loop

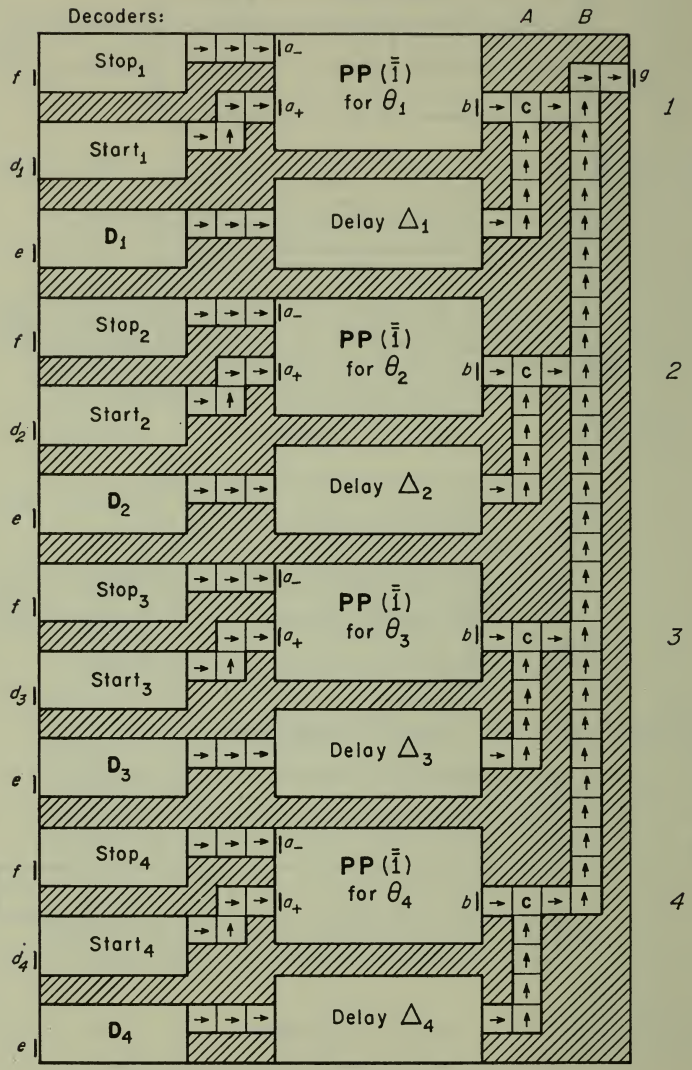


Fig. 53. Static-dynamic converter

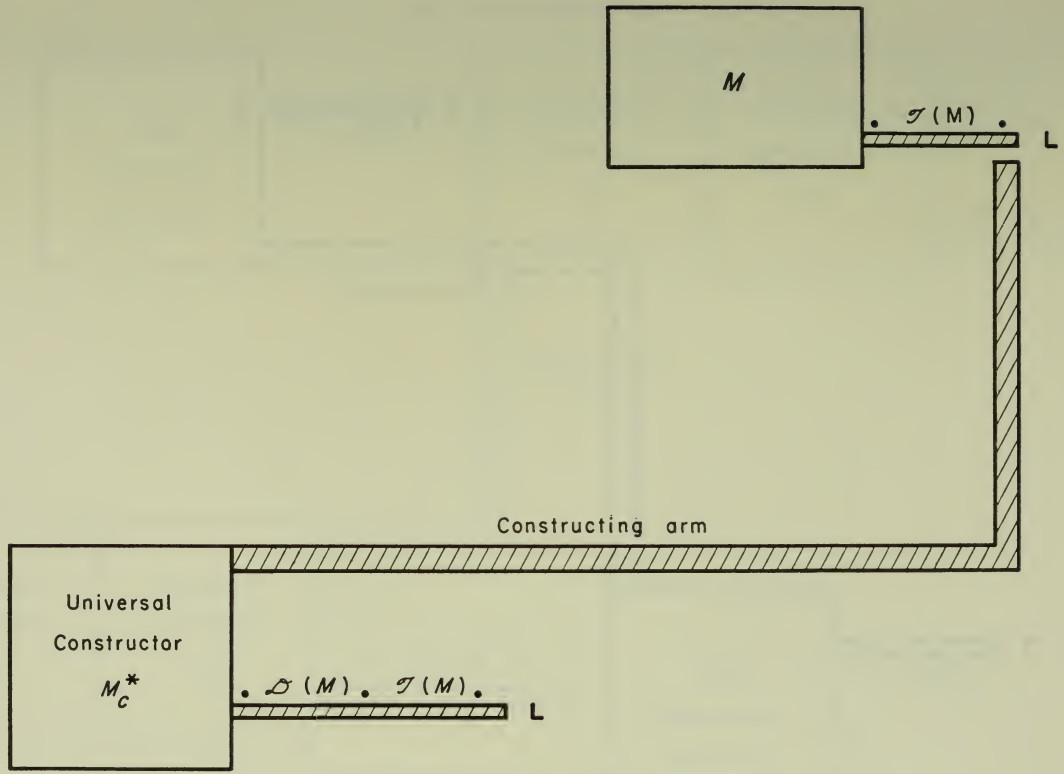


Fig. 54. The universal constructor M_c^*

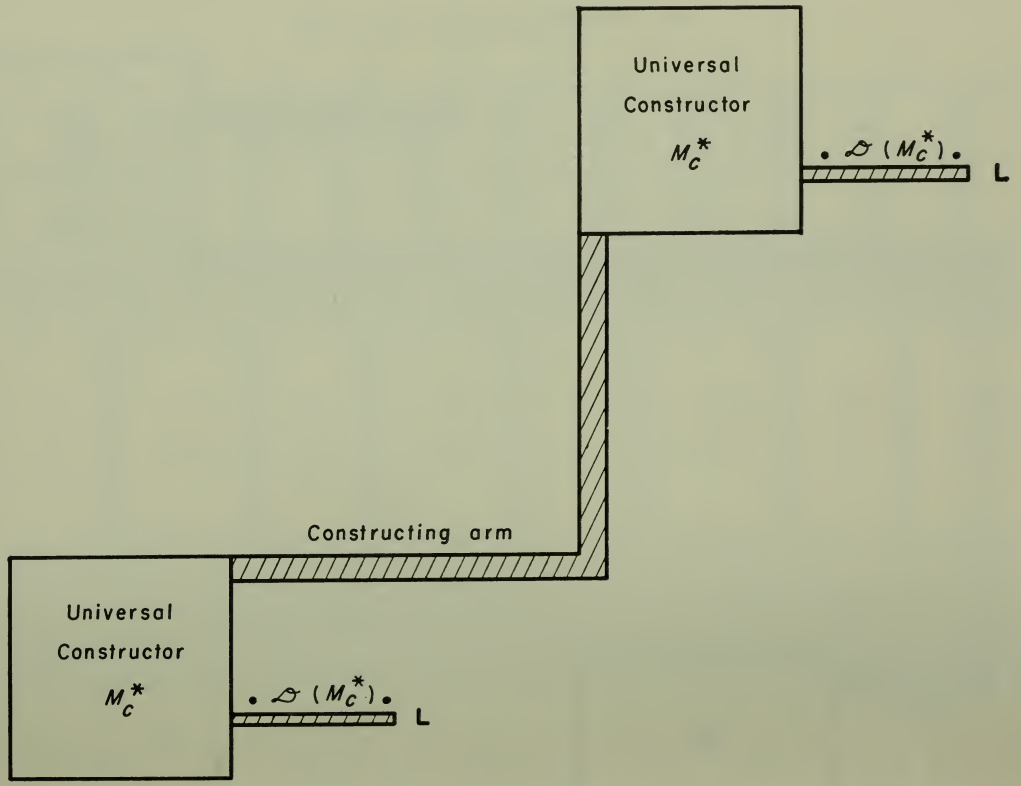


Fig. 55. Self-reproduction

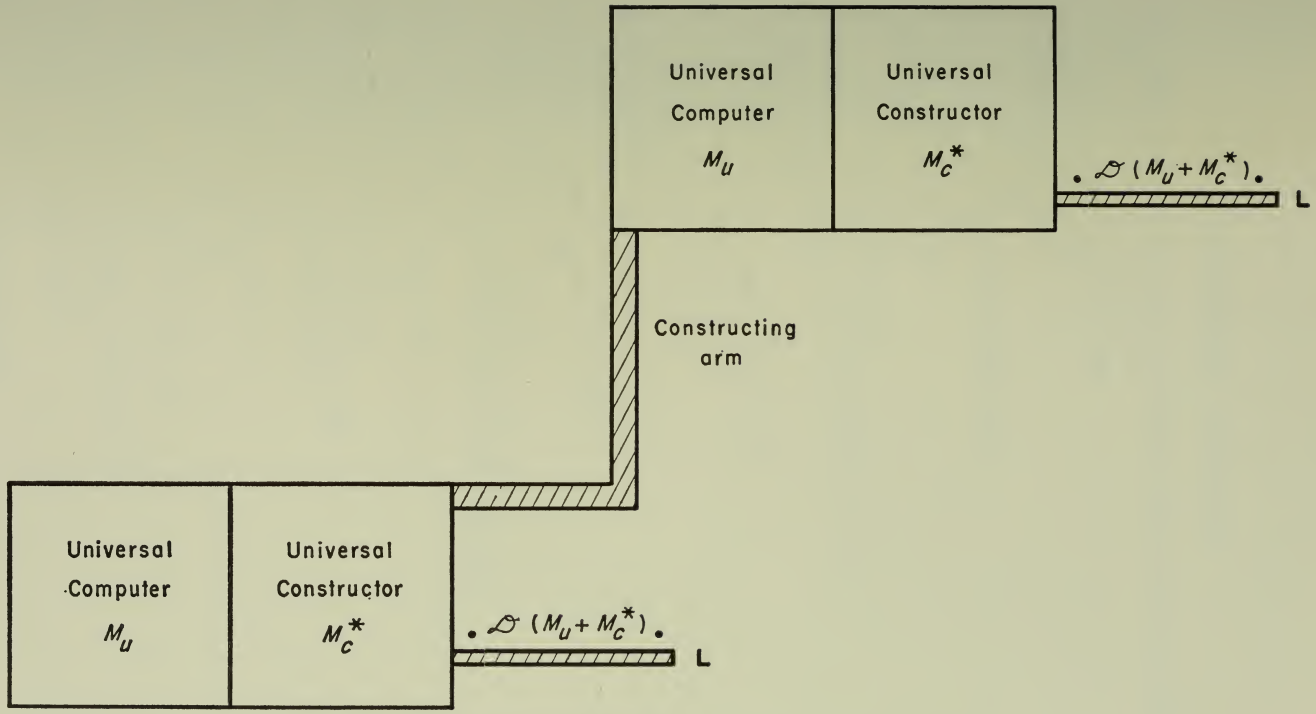


Fig. 56. Self-reproduction of a universal computer-constructor

SYMBOL INDEX

Note: page references are to the text locations where symbols are defined or formally introduced for the first time.

AUTHOR-SUBJECT INDEX