ENGINEERING RESEARCH INSTITUTE
UNIVERSITY OF MICHIGAN
ANN ARBOR

THEORY OF LOGICAL NETS

by

ARTHUR W. BURKS

JESSE B. WRIGHT

Project M828

BURROUGHS ADDING MACHINE COMPANY
DETROIT, MICHIGAN

December, 1952

ensn

UMR0897

TABLE OF CONTENTS

Peter G. Hinman

PREFACE

The present monograph aims to formalize the relation between two-valued logic and digital computing circuits. It will be followed by another in which a detailed application of the present theory will be given.

An informal introduction is included for the benefit of those who are not specialists in the field of symbolic logic.

The authors wish to thank Carl H. Pollmar for many helpful suggestions and Paul Henle for suggesting the basic idea of Theorem 5-5.

Arthur W. Burks

Jesse B. Wright

Engineering Research Institute
University of Michigan
December 23, 1952

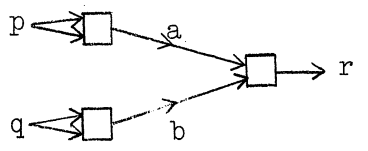
INFORMAL INTRODUCTION

The present monograph deals with some of the theoretical aspects of the application of two-valued symbolic logic to digital computing circuits.

The basic entities of any formal theory are of two kinds: the concepts which constitute the subject matter of the theory, and the principles involving these concepts. We may illustrate this distinction with the example of a two-valued propositional logic, one of the logics studied in this report. The concepts of this logical theory are: propositions which are true (have the value 1) or false (have the value 0); variables ranging over these propositions ($p, q, r, \ldots$); the logical connectives negation (*not*, symbolized by $\sim$), inclusive disjunction (*either* p *or* q *or both*, or $p \vee q$), exclusive disjunction (*either* p *or* q *but not both*, or $p \not\equiv q$), conjunction (*and*, or $\cdot$), implication (*if* p *then* q , or $p \supset q$), and equivalence (p *if and only if* q , or $p \equiv q$). The principles of the theory are such statements as, for all values of p , $p \vee \sim p$; for all values of p, q, r, s, $\sim(p \vee q \vee r \vee s) \equiv (\sim p \cdot \sim q \cdot \sim r \cdot \sim s)$.

The application of a formal theory involves interpreting the basic concepts of the theory so that the principles of the theory become applicable. We will illustrate the application of a two-valued propositional logic to electronic digital computing circuits with reference to the static units of the Burroughs Pulse Control Equipment. Suppose we have a Type 1603A Mixer driving a Type 1901A Inverter. Assign to each signal wire of the circuit a distinct variable; e.g., assign a, b, c, d to the input wires of the Mixer, f to the output wire of the Mixer (which is also the input wire of the Inverter), and g to the output wire of the Inverter. Let the value 1 of a variable mean that the wire it represents is at ground potential and let the value 0 mean that the wire it represents is at -23 volts. Then the Mixer realizes (corresponds to) a four-variable inclusive disjunction, for its behavior is characterized by the equation $f \equiv (a \vee b \vee c \vee d)$. Similarly, the Inverter is a representation of negation, for its behavior is characterized by the equation $g \equiv \sim f$. The relation between the inputs of the circuit and its output is then expressed by $g \equiv \sim(a \vee b \vee c \vee d)$. Since we have characterized the behavior of the Mixer and Inverter by means of the logical concepts of $\sim$ and $\vee$ we may apply the principles of our logic to the analysis of the circuit. Applying the principle that $\sim(p \vee q \vee r \vee s) \equiv (\sim p \cdot \sim q \cdot \sim r \cdot \sim s)$ we find that $g \equiv (\sim a \cdot \sim b \cdot \sim c \cdot \sim d)$, i.e., that the output is at ground potential if and only if all inputs are at -23 volts.

Sheffer showed that all the connectives of the two-valued propositional logic could be defined in terms of a single one, the stroke function $(\sim p \vee \sim q)$. In constructing our theory we found it simpler to work with one connective than with many, so we took as our basic logical element a stroke element, symbolized by $\begin{smallmatrix}p\\q\end{smallmatrix}\!\rightrightarrows\boxed{}\!\rightarrow r$, where $r \equiv (\sim p \vee \sim q)$. Nets of stroke elements can be constructed to represent any expression composed of $p, q, \ldots$ etc., and the logical connectives of the two-valued propositional calculus. For example, $r \equiv (p \vee q)$ is represented by



;

$r \equiv (\sim a \vee \sim b)$ and substituting for a and b we have $r \equiv \sim(\sim p \vee \sim p) \vee \sim(\sim q \vee \sim q)$; by the principle of logic that $\sim(p \vee q) \equiv (\sim p \cdot \sim q)$ we have $r \equiv (\sim\sim p \cdot \sim\sim p) \vee (\sim\sim q \cdot \sim\sim q)$. Using the principle of double negation, we have $r \equiv (p \cdot p) \vee (q \cdot q)$ and hence by the principle that $p \equiv (p \cdot p)$ we have $r \equiv (p \vee q)$. Nets like these are investigated in the present monograph under the name "stroke nets".

A stroke net has no capacity to store or remember information. For this purpose we introduce another element, the delay element $g \longrightarrow \boxed{} \longrightarrow f$. It represents a circuit which receives a standard pulse and gives out a standard pulse after a standard delay; this is normally accomplished by means of delay lines and circuits synchronized by means of uniformly spaced standard pulses produced by a "clock". We may represent the behavior of a delay element by the equations, $f_0 \equiv 0$, $f_{t+1} \equiv g_t$, where t ranges over the discrete time points $0, 1, 2, \ldots,$ and f_t, g_t, etc. take on the values 0 or 1 (e.g., if the wire f has a pulse at t , then $f_t \equiv 1$; otherwise $f_t \equiv 0$). The logical theory of these functions is an extension of two-valued propositional logic, a form of what is called two-valued functional logic. In the present monograph we work mainly with the two-valued logic of these time functions f_t, g_t, etc., and treat the propositional logic as a subcase of this more general logic. This logic is applied to nets made of stroke and delay elements in much the same way the propositional logic is applied to stroke nets. Given a net N we label its junctions with f_t, g_t, etc. and write for each stroke element an equation of the form $f \equiv Sgh$ (abbreviating $f_t \equiv \sim g_t \vee \sim h_t$) and for each delay element an equation of the form $f \equiv Dg$ (abbreviating $f_0 \equiv 0$, $f_{t+1} \equiv g_t$). We then study the net by studying the logical properties of its associated equations.

One of the main questions investigated in this monograph by means of symbolic logic is: To what class of nets do digital computing circuits correspond? We shall summarize briefly our answer to this question. There are nets which are not even logically well-behaved, i.e., whose equations do not have logically consistent and unique solutions. Since no net whose logical behavior is inconsistent or nonunique can be physically realized, we define a class of well-behaved nets (Section 3) which excludes all such cases. However, not all well-behaved nets are physically realizable, so further narrowing of the class of nets is required before we reach a class which corresponds to physical circuits. There are well-behaved nets containing delay elements such that the input state of the delay element at t is determined by its output state at time $t+1$; hence we define a class of deterministic nets (Section 4) which excludes these cases. Finally, there are deterministic nets in which there is a backward passage of causal influence through stroke elements; a class of well-formed nets is defined (Section 5) so as to exclude these. We thus arrive at a class of nets which corresponds very closely to the class of digital computing circuits. The exact correspondence is discussed in Section 6; it is roughly as follows. There are some nets not well-formed which can be physically realized. But for any such net N_1, there is a well-formed net N_2 which can perform all the logical operations of N_1 and which in many cases is at least as small as N_1.

A second major question investigated in the present monograph is: What kinds of logical operations can be performed by various kinds of nets? An answer for each kind of net is given in the corresponding section.

THEORY OF LOGICAL NETS

1. INTRODUCTION

It has been shown by Shannon[‡] and McCulloch and Pitts[‡‡] how two-valued[ǂ] symbolic logic may be employed to characterize the behavior of digital computing circuits; e.g., relay circuits,[ǂǂ] digital electronic computing machines, neuron nets. The purpose of the present paper is to help place the application of two-valued logic to such circuits on a formal and rigorous basis.

We are concerned with two kinds of entities, logical nets and digital computing circuits. A net N usefully represents a circuit C (alternatively, C physically realizes N) when the physical behavior of C is mirrored in an idealized but nevertheless useful way by the logical behavior of N . To realize the purpose mentioned in the preceding paragraph, we will define various kinds of nets, study their formal properties, and discuss the extent to which they can be physically realized.

[‡] Claude E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits," Transactions of the American Institute of Electrical Engineers 57, 713-723 (1938).

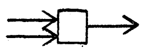
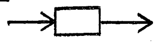
[‡‡] Warren L. McCulloch and Walter Pitts, "A Logical Calculus of the Ideas Immanent in Neuron Activity," Bulletin of Mathematical Biophysics 5-6, 115-133 (1943-44). The diagrams we use are similar to John von Neumann's modification of the diagrams in the above article; cf. Douglas R. Hartree, Calculating Instruments and Machines, pp. 97-111.

[ǂ] Though the present paper is concerned exclusively with two-valued logic, its results are to a certain extent applicable to circuits containing wires having more than two significant states. This application can be made by allowing several wires of a logical net to correspond to a single wire of a physical circuit. E.g., ten of the sixteen different states of four binary net wires can represent ten discrete electrical states of a single circuit wire.

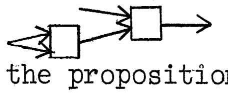
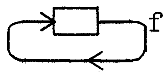
[ǂǂ] The theory developed here is directed mainly to electronic computer circuits, though it is to a degree applicable to relay circuits.

2. PRIMITIVE ELEMENTS

Since the physical components used in digital computing circuits perform either a logical function or a memory† function (or both), the logical analysis of these circuits is facilitated by employing distinct primitive elements for these two functions. Moreover, a single primitive is sufficient for each function, so only two primitive *elements* are required: the stroke element and the delay element.

A *stroke element* consists of a nucleus with two *input wires* and an *output wire* and is symbolized by . Each (net) *wire* has one of two states (0, 1) at each discrete point of time 0, 1, 2, 3, ... Moreover, for each time point t the output wire of a stroke element is in the state 0 if and only if both input wires are in the state 1 ; cf. Sheffer's stroke function. A *delay element* consists of a nucleus, an input wire, and an output wire, and is symbolized by . Each of its wires has one of two states (0, 1) at each discrete point of time, as in the case of the stroke element. The output wire of a delay element is in state 0 at t = 0 , and thereafter it possesses the state possessed by the input wire at the prior point of time.

A *net* is a finite array of elements interconnected so that only the free ends of wires are connected. A *junction* (of a net) is a point common to one or more ends of wires. If there is no arrowhead at a junction it is an *input junction*; otherwise it is an *output junction*. An output junction all of whose output wires are stroke element output wires is a *stroke output junction*; otherwise it is a *delay output junction*.

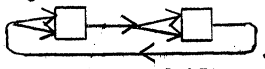
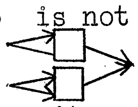
We have chosen these primitives because they possess all the following three properties: (1) nets of them are very useful in studying the behavior of digital computing circuits, (2) they are sufficiently simple to permit convenient logical analysis, and (3) small nets of them correspond to the physical components of digital computers. There are, of course, logically equivalent sets of primitives. The stroke element may be replaced by a conjunctive element and a negative element or by a disjunctive element and a negative element. The stroke element may also be replaced by a material implication element, the equivalent of . Material implication is not by itself a sufficient basis for the propositional calculus, but it and the constant falsehood are;†† the latter may be realized by f in our system, since f(t) ≡ 0 for all t .

† The British "storage" is better than the American "memory" here, since what we call memory components do not by themselves recall or associate information, but only store it.

††Cf. Alonzo Church, *Introduction to Mathematical Logic*, Part I, p. 3.

Finally, the delay element may be replaced (in the presence of the stroke element) by a binary counter with an input for counting and an output in the state 0 or 1 according to the state of the counter, or by a flip-flop with a set input, a reset input, and an output. It is not difficult to show that the equivalent of a delay element can be constructed from either of these and the stroke element.

3. WELL-BEHAVED NETS

For a net N to usefully represent a circuit C there must be a correlation between some of the wires (or sets of wires) of N and some of the wires (or sets of wires) of C such that the states of the designated wires (or sets of wires) of C are represented by the states of the corresponding wires (or sets of wires) of N . Now in any well-behaved circuit the state of every wire at every time t is determinate, and, moreover, all wires joined to the same junction are in the same state. Consequently, we are interested only in nets in which (1) the state of every wire is uniquely determined by the defining properties of the primitive elements and (2) the states of all wires attached to a given junction are the same. We shall call such nets well-behaved (w.b.).[‡]  is not well-behaved because it does not meet the first requirement, while  is not well-behaved because of the second requirement. Physical circuits composed of components which in isolation realize stroke elements and which are connected according to these diagrams would, of course, have some definite behavior (e. g., the wires of such a circuit might oscillate between two states), but this behavior would be controlled by factors we have deliberately omitted from our idealization and would not mirror the logical behavior of the nets.

In this section we shall define precisely the class of well-behaved nets. We begin by correlating to each net a set of logical equations descriptive (in a way to be made clear) of its behavior. Any propositional function of one variable which ranges over the natural numbers and whose values are 0 (false) and 1 (true) will be called (simply) a <u>function</u>. We associate with each junction of a net a distinct function <u>variable</u>. (These variables will also be used to name the junction and each of its wires.) A variable is an <u>input</u>, <u>output</u>, <u>stroke output</u>, or <u>delay output variable</u> according as its junction is an input, output, stroke output, or delay output junction. Finally, we <u>associate</u> with each net N a <u>set of equations</u> E(N) obtained as follows. For each stroke element with input wires g and h and output wire f write the <u>stroke equation</u> $f \equiv Sgh$, and for each delay element with input wire g and output wire f write the <u>delay equation</u> $f \equiv Dg$. In accord with the interpretation of our primitives, the first equation is equivalent to $f_t \equiv \sim g_t \vee \sim h_t$ and the latter is equivalent to $f_0 \equiv 0$ and $f_{t+1} \equiv g_t$, where t ranges over the natural numbers. It is sometimes convenient to have an infinite sequence $E_0(N), \ldots, E_t(N), \ldots$ of sets of equations derived from E(N) as follows: $E_0(N)$ is the result of replacing each stroke equation $f \equiv Sgh$

[‡]It is worth noting that a circuit might be well-behaved for a certain combination of inputs and not for others. This suggests defining a concept of well-behavedness for nets which is relative to a certain class of input sequences. However, it is simpler to work with an absolute sense of well-behaved net and this is what we will do here.

of $E(N)$ by $f_0 \equiv Sg_0h_0$ and each delay equation $f \equiv Dg$ of $E(N)$ by $f_0 \equiv 0$, while $E_t(N)$ (for each $t > 0$) is the result of replacing each $f \equiv Sgh$ by $f_t \equiv Sg_th_t$ and each $f \equiv Dg$ by $f_t \equiv g_{t-1}$.

Given any set of (delay and stroke) equations, a net with which this set is associated may be constructed as follows. Make a junction for each variable occurring in the given set of equations and label it with that variable. For each stroke equation $f \equiv Sgh$ place a stroke element from g and h to f and for each delay equation $f \equiv Dg$ place a delay element from g to f. We shall make the correspondence between nets and sets of equations one-one by arbitrarily identifying all sets of equations differing only with respect to the variables used and with respect to the order of the variables following a stroke operator.

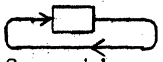
The concept of a well-behaved net introduced informally in the first paragraph of this section may be precisely defined in terms of the uniqueness of the solution of the associated set of equations. A net N with input junctions $a^1, \ldots, a^J$ and output junctions $b^1, \ldots, b^K$ is well-behaved if and only if for each sequence of functions $a^1, \ldots, a^J$ $(J \geq 0)$ there exists a unique sequence of functions $b^1, \ldots, b^K$ $(K > 0)$ such that the a^j's and b^k's satisfy $E(N)$. (Here and hereafter when $I = 0$, $f^1, \ldots, f^I$ is to be interpreted as denoting the null sequence.)

Some auxiliary notions, of use later, will now be defined. Let T be any transformation or mapping from a sequence of functions $f^1, \ldots, f^J$ $(J \geq 0)$ to a function g.[‡] If g is the same for all sequences, T is a constant transformation. (Note that the values of the function g itself are not necessarily the same; e.g., there is a constant transformation to the sequence 010101... .) A transformation may be regarded as an operation on zero or more denumerable sequences of binary digits (0's and 1's), the values of the corresponding functions for $t = 0, 1, \ldots$. Thus the delay transformation D shifts the sequence it operates on one position to the right and places a zero at its beginning. A junction g of a well-behaved net N realizes a transformation T of I arguments $(I \geq 0)$ if and only if among the input junctions $a^1, \ldots, a^J$ of N there are I of them $(a^{h1}, \ldots, a^{hI})$ such that for each sequence of functions $a^1, \ldots, a^J$ the g which satisfies $E(N)$ is equal to $T(a^{h1}, \ldots, a^{hI})$. A junction which realizes a constant transformation is called a constant junction.

A constant transformation to g is periodic if and only if there are integers x and y such that for every integer n, $g_{t+nx+y} \equiv g_{t+y}$. The following theorem shows something concerning the logical power of well-behaved nets.

[‡]If a transformation T_1 of I arguments $(I \geq 0)$ and a transformation T_2 of J arguments $(J > I)$ are such that there exists a sequence of numbers $i_1 < i_2 < \ldots < i_I$ such that for all $f^1, \ldots, f^J$ $T_1(f^{i1}, \ldots, f^{iI}) = T_2(f^1, \ldots, f^J)$, then we shall call T_1 and T_2 the same transformation.

Theorem 3-1: Every constant transformation realized by a well-behaved net is periodic.

Proof: Consider a w.b.n. N with junctions $g^1, \ldots, g^K$ where g^1 is a constant junction. Form $\overline{N}$ from it (with each g^k becoming $\bar{g}^k$) by connecting  to all input junctions of N. Since g^1 realizes a constant transformation, $\bar{g}^1$ realizes the same transformation. Since every junction of $\overline{N}$ is a constant junction, there is a single matrix M satisfying $E(\overline{N})$, where the rows of the matrix give the solution for the $\bar{g}^k$'s and the columns m_0, $m_1, \ldots$ give the state of $\overline{N}$ for times $0, 1, \ldots$ respectively. $\overline{N}$ has at most 2^K states, and hence for some x and y, $m_x = m_{x+y}$. Form a new matrix M' such that for $t < x$, $m'_t = m_t$ and for $t \geq x$, $m'_t = m_{t+y}$. We will prove that $\bar{g}'$ is periodic by showing that $M = M'$. Any matrix satisfies $E(\overline{N})$ if and only if (1) the first column satisfies $E_0(\overline{N})$ and (2) for every t greater than 0 the t-1st and t'th columns satisfy $E_t(\overline{N})$. But if t_1 and t_2 are both greater than zero, $E_{t_1}(\overline{N})$ and $E_{t_2}(\overline{N})$ differ only in their arguments. Hence, condition (2) is equivalent to all pairs of adjacent columns, relabelled for $t = 0$ and $t = 1$, satisfying $E_1(\overline{N})$. Now any pair m'_{t-1} and m'_t $(t > 0)$ satisfies $E_1(\overline{N})$ in this way, for all pairs m_{t-1} and m_t satisfied $E_1(\overline{N})$ in this way, and m'_{x-1}, m'_x is the same pair as m_{x-1}, m_x (since $m_x = m_{x+y}$). Since $\bar{g}^1$ is periodic, g^1 is also. Q.E.D.

A transformation T from $a^1, \ldots, a^J$ to g $(J \geq 0)$ is primitive recursive whenever g can be defined by the operations of primitive recursion and substitution from the successor function, the constant functions, the identity functions, and $a^1, \ldots, a^J$.[†] Something concerning the class of transformations realized by well-behaved nets is given by:

Theorem 3-2: There are primitive recursive transformations not realized by well-behaved nets.

For example, the constant transformation to $f_t \equiv 1$ for t a perfect square, $f_t \equiv 0$ otherwise (i.e., 110010000l...) is clearly a primitive recursive transformation, yet it is a constant transformation which is not periodic and hence by Theorem 3-1 it cannot be realized by any well-behaved net. It is worth noting in this connection that because of its infinite tape a Turing machine[††] is not a net in our sense.

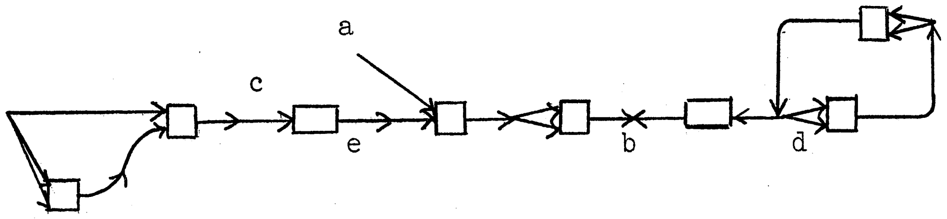
[†] See, e.g., S.C. Kleene, "Recursive Predicates and Quantifiers," Transactions of the American Mathematical Society 53, 42 (1943).

Note that in this definition "function" is used in a sense broader than that defined earlier in this section, since it includes any single-valued function of one or more natural numbers whose values are natural numbers.

[††] Cf. A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungs-Problem," Proceedings of the London Mathematical Society 42, 230-265 (1936-37).

4. DETERMINISTIC NETS

Not all well-behaved nets are physically realizable. Consider N_1 :



It is well-behaved, with $c_t \equiv 1$ (for all t) and hence $b_0 \equiv 0$, $b_{t+1} \equiv a_{t+1}$ and finally $d_t \equiv a_{t+1}$. Thus, junction d "anticipates" the future state of input junction a , and the delay element db performs a kind of inverse delay or predictive operation. Obviously no circuit can perform such a predictive function, for there is no circuit component which can mirror the behavior of the delay element db . More generally, delay elements are intended to represent mechanisms in which signals on input wires cause signals to appear, after a suitable delay, on output wires, and we are interested primarily in nets which allow this interpretation. In other words, we are interested in nets in which all delay elements perform a memory or storage, rather than anticipatory or predictive function. We will call nets of this kind deterministic nets.

Consider next N_2 , formed from N_1 by joining a and e . N_2 is well-behaved, with $e_0 \equiv 0$, $e_{t+1} \equiv 1$, $b_t \equiv e_t$, and $d_t \equiv 1$. Though the (constant) transformation to d can, in this case, be realized by a well-formed net which can in turn be physically realized (cf. Theorem 5-7), it is nevertheless the case that the delay element db performs in N_2 a predictive rather than a memory function, and that hence N_2 is not deterministic. That db does behave in this way may be seen by examining $E_0(N_2)$, $E_1(N_2)$, etc. $E_0(N_2)$ does not determine a unique value of d_0 ; rather, both $d_0 \equiv 1$ and $d_0 \equiv 0$ satisfy $E_0(N_2)$. It is only when $E_1(N_2)$ is considered that the possibility of $d \equiv 0$ is excluded. This analysis leads directly to one definition of deterministic net. Let N have input junctions $a^1, \ldots, a^J$ $(J \geq 0)$ and output junctions $b^1, \ldots, b^K$. First definition: N is <u>deterministic</u> if and only if for each sequence $a_0^1, \ldots, a_0^J; \ldots; a_t^1, \ldots, a_t^J$ $(t \geq 0)$ there exists a unique sequence $b_t^1, \ldots, b_t^K$ satisfying the union of $E_0(N), \ldots, E_t(N)$.

Where a net with input junctions is involved, this definition means roughly that the states of the input junctions for the past and the present determine the states of the output junctions for the present. An alternative conception of determinism is that of the state of the net at the immediate past $(t-1)$ and the state of the input junctions at the present (t) determining the state of the output junctions at the present. Since this conception is also useful, we will give a second definition of determinism and then prove that the two definitions are equivalent. Second definition: N is <u>deterministic</u> if and only if both of the following hold: (1) for each $a_0^1, \ldots, a_0^J$ there exists a unique $b_0^1, \ldots, b_0^K$ such that $a_0^1, \ldots, a_0^J; b_0^1, \ldots, b_0^K$ satisfies $E_0(N)$; and (2) if $t > 0$ then for each $a_0^1, \ldots, a_0^J; \ldots; a_{t-1}^1, \ldots, a_{t-1}^J$ and $b_0^1, \ldots, b_0^K; \ldots; b_{t-1}^1, \ldots, b_{t-1}^K$ satisfying the union of $E_0(N), \ldots, E_{t-1}(N)$ and each $a_t^1, \ldots, a_t^J$, there exists a unique $b_t^1, \ldots, b_t^K$ such that $a_{t-1}^1, \ldots, a_{t-1}^J; a_t^1, \ldots, a_t^J; b_{t-1}^1, \ldots, b_{t-1}^K; b_t^1, \ldots, b_t^K$ satisfies $E_t(N)$. (If $J = 0$ various of the sequences involved are to be taken to be the null sequence as before.)

<u>Theorem</u> 4-1: The two definitions of "determinism" are equivalent.

<u>Proof</u>: That the second definition implies the first may be shown by a simple induction. The proof that the first definition implies the second is as follows. Let N be deterministic by the first definition. Then $a_0^1, \ldots, a_0^J; \ldots; a_{t-1}^1, \ldots, a_{t-1}^J$ determines a unique $b_0^1, \ldots, b_0^K; \ldots; b_{t-1}^1, \ldots, b_{t-1}^K$ satisfying the union of $E_0(N), \ldots, E_{t-1}(N)$; and both of these sequences plus $a_t^1, \ldots, a_t^J$ determine a unique $b_t^1, \ldots, b_t^K$ satisfying the union of $E_0(N), \ldots, E_{t-1}(N)$ plus $E_t(N)$. But since $E_t(N)$ involves only the a^j's and b^k's at $t-1$ and t , it follows that the $a_{t-1}^1, \ldots, a_{t-1}^J; a_t^1, \ldots, a_t^J; b_{t-1}^1, \ldots, b_{t-1}^K$ determines a unique $b_t^1, \ldots, b_t^K$ satisfying $E_t(N)$. Q.E.D. This last argument makes use of the fact that long-term memory (i.e., memory lasting more than one unit of time) in a net is the result of a succession of unit delays. This would not be the case, for example, if we had an additional two-wire primitive element capable of n units of delay, $n > 1$; if that were the case the two definitions would not be equivalent.

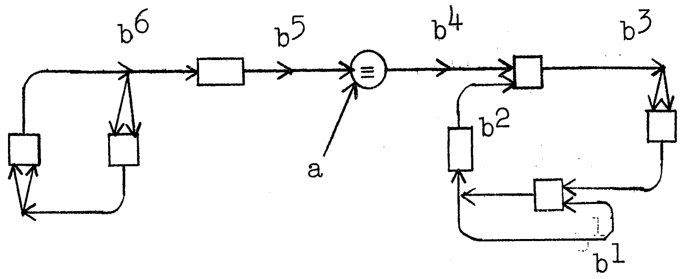
The class of deterministic nets is a subclass of the class of well-behaved nets:

<u>Theorem</u> 4-2: Every deterministic net is well-behaved.

The proof of this is obvious from the definitions.

We shall next define some concepts which will be of use in characterizing nondeterministic as well as other kinds of w.b.n. Any junction to which two or more output wires are connected is a <u>multiple junction</u>. A junction f <u>directly drives</u> a junction g if and only if f is the input of a stroke element whose output is g . (Note that the input of a delay element does not directly drive its output.) f <u>drives</u> g if and only if there exists a sequence $f^1, f^2, \ldots, f^I$ such that f^1 is f , f^I is g , and f^i directly drives f^{i+1} for $i < I$. Note that the driving relation is transitive: if f drives g and g drives h then f drives h . A <u>stroke cycle</u> is a sequence of junctions $f^0, \ldots, f^{I-1}$ such that $f^{i \text{ Mod } I}$ directly drives $f^{(i+1) \text{ Mod } I}$.

It will be observed that N_1 and N_2 contain multiple junctions, stroke cycles, and delay elements. Not every nondeterministic w.b.n. contains a multiple junction, however, as the following net proves:



where the circle with the equivalence sign in it represents a net (constructable from stroke elements) realizing $b^4 \equiv (b^5 \equiv a)$. $b^1_t \equiv 1$ and hence $b^2_{t+1} \equiv 1$ and also $b^3_t \equiv 1$. These two equations imply that $b^4_{t+1} \equiv 0$, which in turn implies that $b^5_{t+1} \equiv \sim a_{t+1}$. Since $b^6_t \equiv b^5_{t+1}$, it follows that $b^6_t \equiv \sim a_{t+1}$, and so the net is nondeterministic. On the other hand,

<u>Theorem</u> 4-3: Every nondeterministic w.b.n. contains a stroke cycle and a delay element.

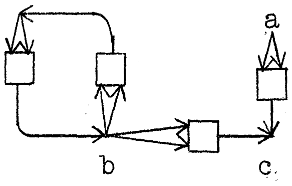
It will be simpler to give the proof of the first part in the next section (immediately following the proof of Theorem 5-4). The second half may be stated alternatively: every w.b. <u>stroke net</u> (a net constructed exclusively of stroke elements) is deterministic. The proof of it is as follows. Let N be a w.b. stroke net with input junctions $a^1, \ldots, a^J$ and output junctions $b^1, \ldots, b^K$. Then E(N) contains no delay equations, all functions in $E_t(N)$ have t as an argument (never t-1), and hence the state of the net at t cannot place any restrictions on the state of the net at t-1 (and vice versa). Therefore, for each $a^1_t, \ldots, a^J_t$ there is a unique $b^1_t, \ldots, b^K_t$ satisfying $E_t(N)$ [since for each $a^1, \ldots, a^J$ there is a unique $b^1, \ldots, b^K$ satisfying E(N)]. Q.E.D. The basic idea of this proof can be employed to prove a stronger result concerning w.b. stroke nets, namely:

<u>Theorem</u> 4-4: All transformations realized by w.b. stroke nets are truth-transformations.

The concept of truth-transformation is defined as follows. Some transformations T (from $a^1, \ldots, a^J$) to g are equivalent to a sequence of mappings T_0, T_1, ..., each T_t being from a sequence of J truth-values to a single truth-value (0,1) . If all T_t are identical, T is said to be a <u>truth-transformation</u>. The result that all transformations realized by w.b. stroke nets are truth-transformations is a justification of an earlier statement (Section 2) that the stroke element performs only a logical function; a w.b. stroke net has no memory, its state at a given time never depends on its state at a prior time and never influences its state at a later time.

5. WELL-FORMED NETS

We referred earlier (Section 2) to physical components which perform a logical rather than memory function and have represented these in our system by stroke nets. In actual fact, these components function much the same as the components realizing delay elements in that signals on their input wires cause signals to appear, after a certain delay, on output wires. The differences between the temporal lag inherent in a logical component and that in a memory component are that the former is generally smaller than the latter and generally serves no useful purpose. On this account, and in the interest of logical simplicity, our theory includes a temporal delay only in the delay element. As a consequence, however, we should not expect the requirement of determinism to eliminate all nets in which there is a backward passage of causal influence through stroke elements. Thus in the w.b.n.



information "flows" from the output to the inputs of the stroke element bc , yet this net is deterministic, with $a_t \equiv \sim c_t \equiv b_t$. Since no component can mirror the logical behavior of the stroke element bc (though of course the transformation it realizes is easily realized physically), not all stroke elements in deterministic nets perform functions which can be duplicated by physical components.

In the present section we will define a class of nets, called well-formed nets (w.f.n.), which excludes the example just discussed. For reasons given in the next section, this is the most useful class of nets for the study of the behavior of digital computing circuits. There are w.f.n. representing, with various degrees of utility and directness, systems of neurons (in which the time intervals are of the order of one millisecond), some systems of electromechanical relays (in which the time intervals are of the order of milliseconds), and many electronic computing machines composed of vacuum tubes, crystal rectifiers, acoustic delay lines, and electrostatic storage tubes. In particular, w.f.n. characterize very closely the behavior of that type of electronic digital computer whose action is governed by a "clock" which is the source of equally-wide equally-spaced standard pulses (with the time interval between pulses being of the order of one microsecond).[‡] In fact, our theory is especially directed towards such computers. In this case, a 1

[‡]Cf. Hartree, _op. cit._

represents the occurrence of a pulse and a 0 the absence of a pulse, or vice versa.

The class of well-formed nets is defined recursively as follows, with the understanding that no net is w.f. unless its being so follows from these rules.

(1) (a) [Stroke rule] A stroke element is w.f.
(b) [Delay rule] A delay element is w.f.

(2) Assume N_1 and N_2 are disjoint w.f.n. with junctions $f^1, \ldots, f^J$ and $g^1, \ldots, g^K$ respectively.

(a) [Juxtaposition rule] The juxtaposition of N_1 and N_2 is w.f.

(b) [Cascade rule] The result of joining junctions $f^{q1}, \ldots, f^{qI}$ of N_1 to distinct input junctions $g^{p1}, \ldots, g^{pI}$ of N_2 respectively is w.f.

(c) [Input connection rule] The result of joining input junctions f^p and f^q of N_1 is w.f.

(d) [Cycle rule] If all the wires connected to f^p of N_1 are delay element input wires, then the result of joining any f^q of N_1 to f^p is w.f.

A w.f.n. N may conveniently be studied by means of E(N) . E(N) is unitary if and only if no variable has more than one left-hand occurrence (i.e., to the left of the equivalence sign in a stroke or delay equation) in E(N) . E(N) is regular if and only if there is an ordering (called a regular ordering) of E(N) such that for all f if f has a right-hand stroke equation occurrence in E(N) then there is no left-hand occurrence of f in that or any later equation. The regular ordering of a regular set of equations can be obtained by using the concept of rank. Ranks may be assigned to certain of the junctions and corresponding equations of any net as follows. First, assign the rank 0 to all input junctions and all delay output junctions to which no stroke element output wires are connected. Then iterate as long as possible the general step: Assign the rank r to all junctions f such that some junction directly driving f has rank r-1 and every junction directly driving f has a rank $\leq$ r-1 . Finally, to every equation $f \equiv$ —— which is such that the junction f has a rank assign the rank of f .

Theorem 5-1: N possesses no multiple junctions if and only if E(N) is unitary; and N contains no stroke cycle if and only if E(N) is regular.

Proof: The first half is obvious. The "only if" part of the second half follows from the fact that the set of equations for a stroke cycle is not regular and hence any set of equations containing them is not regular. That E(N) is regular if N contains no stroke cycles may now be proved as follows. We prove first that since N contains no stroke cycles every junction of N is assigned a rank. Assume some junction f of N has no rank. Then there must be a junction g^1 which directly drives it which has no rank. In turn, there must be a g^2 directly driving g^1 and without rank, ad infinitum. But there are only a finite number of junctions in N , so some variable g^i in the sequence $g^1, g^2, \ldots$ must occur twice in this sequence. Hence N contains a stroke cycle, contrary to our assumption. Since every junction of N is assigned a rank, every equation of E(N) is assigned a rank, and these equations may be arranged in order of ascending rank. We prove finally that this ordering is regular. Consider any equation of the form $f \equiv Sgh$, where g is of rank r . Then f is of rank $\geq r+1$; any equation of the form $g \equiv$ —— is of rank r and hence must precede $f \equiv Sgh$. Q.E.D.

We next prove

<u>Theorem</u> 5-2: N is w.f. if and only if E(N) is both unitary and regular.

<u>Proof</u>: We prove first that if N is w.f. it contains no multiple junctions and no stroke cycles. A stroke element or delay element by itself (cf. the stroke and delay rules) contains no multiple junctions or stroke cycles. Moreover, if N_1 and N_2 contain no multiple junctions or stroke cycles, the result of combining them by any of the four rules of combination will not contain any multiple junctions or stroke cycles. It follows by Theorem 5-1 that if N is w.f., E(N) is both unitary and regular. We will prove the converse by giving a stepwise process for constructing N on the basis of S , a regular ordering of E(N) . Let S_i be that subsequence of S which consists of the first i equations of S . The net associated with S_1 is w.f. (by rules 1, 2c, and 2d) since E(N) is regular. Assume now that N_1^i is formed from S_{i-1} and is w.f. and consider the i'th equation of S . The element associated with it may have two input wires which are paired or not; in the former case use the input connection rule to get N_2^i ; in the latter case take the element itself to be N_2^i . N_2^i may have no input wire connected to a junction of N_1^i in N or one or more such; in the former case juxtapose N_2^i and N_1^i ; in the latter case cascade N_1^i onto N_2^i . Finally, consider the output f of N_2^i . Since S_i is regularly ordered and unitary, the only occurrences of f other than the left-hand occurrence in the last equation of S_i are right-hand delay equation occurrences in the equations of S_i . Hence the output of N_2^i can be connected into the net (if necessary) by the cycle rule. Q.E.D.

<u>Theorem</u> 5-3: E(N) is regular if and only if every junction of N has a rank.

<u>Proof</u>: In Theorem 5-1 it was shown that if every junction of N has a rank then E(N) is regular; and also that if E(N) is regular, then N contains no stroke cycles and that if N contains no stroke cycles then every junction of N has a rank. Q.E.D.

Thus the following four properties of N are equivalent: (1) N is w.f., (2) N has no multiple junctions and no stroke cycles, (3) N has no multiple junctions and every junction of N has a rank, and (4) E(N) is both unitary and regular, and any one may be employed as a criterion of a w.f.n. The formation rules are naturally and simply applied when a net is being constructed. Given a complicated net, however, it may be difficult to determine whether or not it can be constructed by the rules. In this case a test procedure based on criterion (3) is useful. Label all junctions of rank 0, stopping if any are multiple junctions (since then N is not w.f.). Then iterate the following step as long as possible: label all junctions of rank r, stopping if any are multiple junctions. If this procedure terminates without a multiple junction being discovered, the N is w.f. if and only if all junctions have been assigned ranks (by Theorem 5-3).

<u>Theorem</u> 5-4: Every w.f.n. is deterministic, and hence well-behaved.

(Note that the example given at the beginning of this section shows that the converse does not hold.)

<u>Proof</u>: The determinism of a w.f.n. follows from the fact that (in the presence of unitariness) regularity is stronger than determinism, allowing not only sequential computation with respect to time but also sequential computation at a given time. For each t the state of a w.f. N may be computed by proceeding through a regular ordering S of E(N) as follows. If the equation at hand is a delay equation $f \equiv Dg$, then for $t = 0$, $f_0 \equiv 0$ and for $t > 0$, $f_t \equiv g_{t-1}$, where g_{t-1} is known from the state of the net at $t-1$. If the equation at hand is a stroke equation $f \equiv Sgh$, then, since S is in regular order, g and h have no left-hand occurrences in that or any later equations of S. Hence g (h) either has a left-hand occurrence in an earlier equation of S, in which case g_t (h_t) is already known from the preceding computation; or g (h) has no left-hand occurrence in any equation, in which case it is an input variable and g_t (h_t) is given. Since E(N) is unitary, this procedure gives a unique value for each f_t. Hence N is deterministic and by Theorem 4-2 it is also w.b. Q.E.D. An alternative way of looking at the computation is in terms of rank: the states of all junctions of rank 0 may be determined, then those for rank 1, etc.

By a procedure similar to that just used, we will now prove the first half of Theorem 4-3, namely that every nondeterministic w.b.n. contains a stroke cycle. This may be stated alternatively as: Every w.b.n. containing no stroke cycles is deterministic. The proof is as follows. Let N be a w.b.n. with no stroke cycles. By Theorem 5-1 E(N) is regular. Hence the behavior of N can be computed in the manner indicated in the last paragraph, and since N is w.b. the result will be unique.

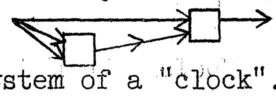
We prove next a theorem concerning the nature of the subnets of a w.f.n. N_1 is a <u>subnet</u> of N if and only if N_1 may be formed from N by a succession (perhaps null) of the following operations: (1) separating a wire from a junction, and (2) deleting an element.

Theorem 5-5: N is w.f. if and only if all the subnets of N are w.b.

Proof: If N is w.f., it contains no multiple junctions or stroke cycles, and hence neither do any of its subnets. Therefore each of its subnets is w.f. and hence w.b. We prove next that if N is not w.f. it contains a subnet which is not w.b. There are two cases to consider, according to whether N contains a multiple junction or not. (1) Suppose N contains a multiple junction f such that $f \equiv$ — and $f \equiv \ldots$ are in $E(N)$. Separate the input wires of the elements associated with each of the equations from their junctions. The result is not w.b. (2) Suppose N contains no multiple junctions but does contain a stroke cycle $f^0, \ldots, f^{I-1}$. We may assume without loss of generality that the associated sequence of equations is $f^{I-1} \equiv Sf^{I-2}\, g^{I-2}, \ldots, f^0 \equiv Sf^{I-1}\, g^{I-1}$. If any of these equations is of the form $p \equiv Sqq$, separate one of the input wires of the corresponding stroke element from its junction, making it a net input. Then if any g^i input wire of an element corresponding to one of these equations is connected to an output junction, separate it from that junction so that it is a net input. The result is not w.b., as may be shown by applying 1's to all input junctions. Q.E.D.

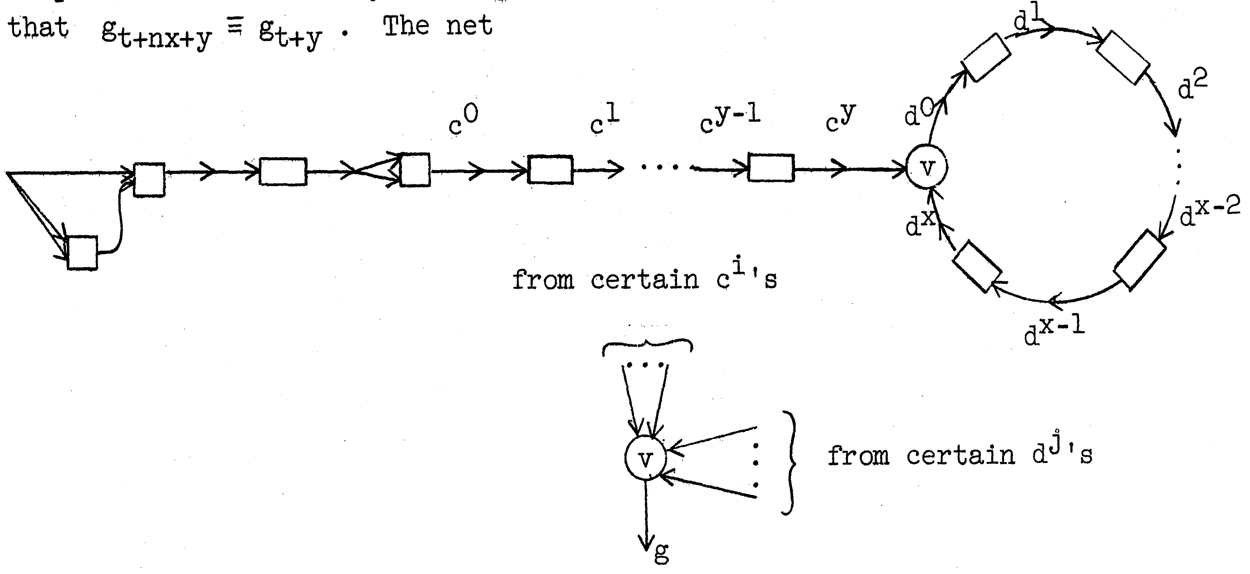
We next state some theorems concerning the transformations realized by w.f.n.

Theorem 5-6: A transformation is realizable by a w.f. stroke net if and only if it is a truth-transformation.

Proof: We have already proved (Theorem 4-4) that all transformations realized by w.b. (and hence by w.f.) stroke nets are truth-transformations, so it need only be proved that all truth-transformations can be realized by w.f. stroke nets. Since Sheffer's stroke function is a sufficient primitive for defining all truth-functions (in the logical sense), it is a sufficient primitive for defining all truth-transformations. The process of definition involves substitution and the use of the same variable in more than one argument place; these are mirrored by the cascade rule and the input connection rule, respectively. Q.E.D. The last half of the theorem shows that (in the presence of the formation rules) a stroke element is a sufficient primitive for our purposes. It follows from the definition of truth-transformation that there are only two constant truth-transformations: $g_t \equiv 1$ and its negation $h_t \equiv 0$. The former is realized by the w.f.n.  g which is of interest because it is a realization in our system of a "clock". (Note the fact—mentioned earlier in this section—that our theory is especially applicable to circuits synchronized by pulses from a standard clock.)

Theorem 5-7: A constant transformation is realized by a w.f.n. if and only if it is periodic.

Proof: The "only if" part of this theorem follows from Theorems 3-1 and 5-4. We prove the converse by showing how to construct a net realizing any g such that $g_{t+nx+y} \equiv g_{t+y}$. The net



is a schematic construction of the desired net, where $\begin{matrix} f^1 \\ \vdots \\ f^J \end{matrix} \rightarrow \textcircled{v} \rightarrow$ represents a J-input disjunctive net, readily constructed from stroke elements. Note that $c_0^0 \equiv 1$, $c_{t+1}^0 \equiv 0$, and in general $c_t^i \equiv 1$ if and only if $t = i$, while $d_t^j \equiv 1$ if and only if $t = j + nx + y$. For a particular constant transformation to g this schema may be converted into a net realizing this transformation as follows. If $g_t \equiv 1$ for any t less than y , connect c^t to one of the inputs of the disjunctive net whose output is g ; if $g_t \equiv 1$ for any t as large as y and less than $x+y$, connect d^t to one of the inputs of the disjunctive net whose output is g . g is the desired junction. Q.E.D.

Theorem 5-8: Every transformation realized by a w.f.n. is primitive recursive.

The proof is too long to be included here (it is given in the appendix), but the essential idea is as follows. Let N be w.f. with input junctions a^1, ..., a^J and output junctions b^1, ..., b^K . Since the behavior of N can be computed in the manner indicated in the proof of Theorem 5-4, it is possible to define a mathematical function F (from natural numbers to natural numbers) which is primitive recursive with respect to the a^j's and is such that

$$F(0) = 2^{b_0^1} \cdot 3^{b_0^2} \cdots p_{K-1}^{b_0^{K-1}} \cdot p_K^{b_0^K} \cdot p_{K+1}^{b_1^1} \cdots p_{2K-1}^{b_1^{K-1}} \cdot p_{2K}^{b_1^K}$$

$$F(1) = 2^{b_0^2} \cdot 3^{b_0^3} \cdots p_{K-1}^{b_0^K} \cdot p_K^{b_1^1} \cdot p_{K+1}^{b_1^2} \cdots p_{2K-1}^{b_1^K} \cdot p_{2K}^{b_2^1} ,$$

etc., where p_k is the k-th prime, in order of magnitude. Each b^k can then be defined primitive recursively in terms of F as follows: b_t^k equals the first exponent of $F(tK + k - 1)$.

6. W.F.N. AND DIGITAL COMPUTING CIRCUITS

In this section we will discuss the realization of w.f.n. by digital computing circuits. Consider first the fact that both the delay element and the stroke element (or small nets of it) are realized by physical components in which the input wire state(s) causally determine the output wire state. The output of the physical realization of a delay or stroke element in isolation has the property that its state can be causally determined by the state(s) of its input wire(s) and the formation rules for w.f.n. are such that every output wire of a circuit realizing a w.f.n. has this property. Thus the backward passage of causal influence through stroke elements (cf. the first net discussed in the preceding section) never occurs in a w.f.n. Hence, as far as these considerations are concerned every w.f.n. is physically realizable. There are, however, certain aspects of our idealization of a stroke element which require this conclusion to be restricted. In particular, though the delay of a component realizing a simple stroke net is generally small compared to the basic unit of time of the system, the same does not hold for circuits composed of such components. A practical way of dealing with this point is to construct circuits which perform fairly simple logical transformations and then feed the result into a delay line (as well as into power amplifying and retiming circuits), the temporal lag of the line being such that the total delay is one or a few units of time. This involves placing an upper limit on the allowable rank of a junction and on the number of junctions which drive a given junction. These restrictions could be removed from the net theory by taking as a primitive element, e.g., a stroke element driving a delay element, but the additional complications would not be worth the gain. In any event, there is an overall size limitation on the realizability of w.f.n.

The discussion of Section 4 showed that if a circuit is represented by a well-behaved net that net is deterministic. Consider now the question: Are there any circuits represented by deterministic nets which could not be adequately represented by w.f. nets? To keep the range of application of the logic of nets as broad as possible, we have left the concept of representation (and its converse, realization) somewhat indefinite. It is sufficiently broad that some so-called "static" circuits and some mixed static and pulse circuits, as well as some pulse ("dynamic") circuits can under suitable limitations be represented by w.f.n. The answer to the above question depends on the particular interpretation made and hence we will not attempt to give a general answer here; we will, however, make a few relevant comments. Engineers sometimes connect the outputs of two or more physical components together; this is a way of realizing logical disjunction and is better represented in our system by stroke nets realizing logical disjunction than by nets containing multiple junctions. Also, feedback loops are sometimes employed for memory purposes (as in the "flip-flop"), but these are better represented in our system by means of delay elements rather than by stroke cycles.[‡]

On the other hand, consider the following example of a deterministic net not w.f. which may be physically realized in one sense of physical realization. Start with a single physical component realizing $f \equiv (g \cdot \sim h)$ and connect its inputs together so that $f \equiv (g \cdot \sim g) \equiv 0$. Its physical behavior would probably be unaffected by connecting its output back to its input. Note, however, that this connection, which introduces a stroke cycle in the corresponding net, is pointless, since it neither increases the number of transformations realized nor decreases the number of elements in the net. Thus this example gives rise to the following two theoretical questions which are worth investigating: (I) Can deterministic nets not w.f. realize transformations not realized by w.f.n.? and (II) Can they realize some transformations more efficiently? The answer with regard to (I) is given by:

<u>Theorem</u> 6-1: Every transformation realized by a deterministic net is realized by a w.f.n.

<u>Proof</u>: Let N be deterministic with input junctions $a^1, \ldots, a^I$, delay output junctions $b^1, \ldots, b^J$, and stroke output junctions $c^1, \ldots, c^K$; and let $\overline{N}$ be the desired w.f.n. whose junctions include the corresponding junctions $\overline{a}^1, \ldots, \overline{a}^I$; $\overline{b}^1, \ldots, \overline{b}^J$; and $\overline{c}^1, \ldots, \overline{c}^K$. We define a sequence of truth-values (0,1) $x_1, \ldots, x_J$ to be <u>admissible</u> [relative to E(N)] if and only if it satisfies either of the following two conditions (cf. the second definition of determinism): (1) it consists of all 0's , and (2) there is some $t > 0$ and some sequence $a^1_0, \ldots, a^I_0; b^1_0, \ldots, b^J_0; c^1_0, \ldots, c^K_0;; \ldots;; a^1_t, \ldots, a^I_t; b^1_t, \ldots, b^J_t; c^1_t, \ldots, c^K_t$ satisfying the union of $E_0(N), \ldots, E_t(N)$ and such that for each j , $b^j_t = x_j$. Let $E^c_t(N)$ consist of all equations of $E_t(N)$ of the form $c^k_t \equiv$ —— (i.e., of the form $c^k_t \equiv S \ldots$). Since N is deterministic it then follows that for each $a^1_t, \ldots, a^I_t$ and each admissible $b^1_t, \ldots, b^J_t$ there is a unique $c^1_t, \ldots, c^K_t$ such that $a^1_t, \ldots, a^I_t; b^1_t, \ldots, b^J_t; c^1_t, \ldots, c^K_t$ satisfies $E^c_t(N)$. Now define for each c^k_t a mapping T^k_t as follows: for each $a^1_t, \ldots, a^I_t$ and each admissible $b^1_t, \ldots, b^J_t$ $T^k_t(a^1_t, \ldots, a^I_t; b^1_t, \ldots, b^J_t)$ is that value of c^k_t which satisfies $E^c_t(N)$; for each $a^1_t, \ldots, a^I_t$ and each non-admissible $b^1_t, \ldots, b^J_t$ $T^k_t(a^1_t, \ldots, a^I_t; b^1_t, \ldots, b^J_t) = 0$. Since each $E^c_t(N)$ is of the same form (i.e., a difference in t involves only a difference in the arguments), we may define for every c^k a truth-transformation T^k equivalent to the sequence of mappings $T^k_0, T^k_1, \ldots$. The net $\overline{N}$ may now be constructed as follows. For each b^j pick an equation of E(N) of the form $b^j \equiv Df$ and connect a delay element from $\overline{f}$ to $\overline{b}^j$ in

[‡]By altering the defining equations of a stroke-element we can obtain a "stroke net" which performs a memory function. For example, a net with inputs s (set) and r (reset) and junctions f and g such that $f \equiv (\sim s \cdot \sim g)$ and $g \equiv [\sim f \cdot (\sim r \vee s)]$ is a representation of a flip-flop under the stipulation that if $s_t \equiv r_t \equiv 0$ then (1) if $t = 0$, then $f_t \equiv 0$ and (2) if $t > 0$, then $f_t \equiv f_{t-1}$.

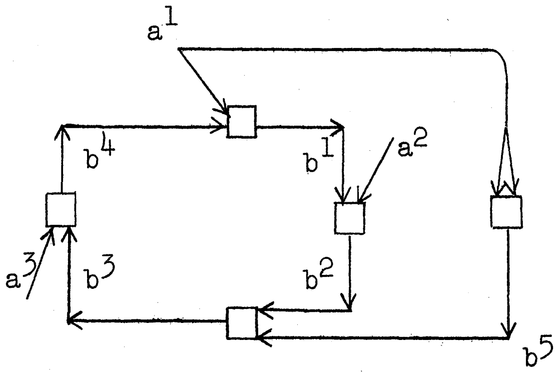
$\bar{N}$. For each $\bar{c}^k$ of $\bar{N}$ construct a w.f. stroke net (cf. Theorem 5-6) from the $\bar{a}^i$'s and $\bar{b}^j$'s realizing T^k . The transformations realized by the $\bar{b}^j$'s and $\bar{c}^k$'s in $\bar{N}$ are the transformations realized by the b^j's and c^k's of N respectively, and since no stroke cycles and no multiple junctions were created in the formation of $\bar{N}$, $\bar{N}$ is w.f. Q.E.D.

Consider now question (II). This involves the concept of simplicity. For nets composed of stroke and delay elements a rough measure of simplicity is given by the number of each. Some results concerning this rough measure will now be stated. Let us arbitrarily define N_1 to be <u>as simple as</u> N_2 whenever N_1 has no more stroke elements and no more delay elements than does N_2 . It should be remembered that logically equivalent primitives are not necessarily equivalent where simplicity is involved, so our subsequent theorems on simplicity do not necessarily hold for alternative sets of primitive elements logically equivalent to the set consisting of the stroke and delay elements (see Section 2). Question (II) can now be more precisely stated: Given a deterministic net N not w.f., is there a w.f.n. as simple as N which realizes the same transformations as N ?

The answer is in the negative, for:

<u>Theorem</u> 6-2: There is a deterministic net N such that no w.f.n. realizing all the transformations realized by N is as simple as N .

<u>Proof</u>: Let N be:



N contains the stroke cycle b^1, b^2, b^3, b^4 and hence is not w.f., but by Theorem 4-2 it is deterministic. It is easily verified that N realizes the distinct transformations

$$b^1 \equiv (\sim a^1 \vee a^3)$$
$$b^2 \equiv (\sim a^2 \vee a^1 \sim a^3)$$
$$b^3 \equiv a^1 \vee a^2$$
$$b^4 \equiv \sim a^3 \vee \sim a^1 \sim a^2$$
$$b^5 \equiv \sim a^1$$

Consider now a w.f.n. N_1 with (at least) junctions b^1, b^2, b^3, b^4, and b^5, realizing the corresponding transformations. For each b^k, $1 \leq k \leq 5$ there must be exactly one equation in $E(N_1)$ of the form $b^k \equiv S$ —— . Replace $b^5 \equiv S$ —— by $b^5 \equiv Sa^1a^1$; this modification of N_1 does not change the total number of stroke elements in N_1 nor the number of transformations realized by N_1 . Let the minimal rank of b^1, b^2, b^3, b^4 of N_1 be R and let b^x be one of the junctions b^1, b^2, b^3, b^4 of N_1 of this minimal rank. None of the transformations b^1, b^2, b^3, b^4 is equivalent to Sfg where f and g are a^1, a^2, a^3 or b^5, and hence b^x cannot be realized from a^1, a^2, a^3, b^5 by less than two stroke elements. Consequently, there is a junction b^6 directly driving b^x which is different from a^1, a^2, a^3, and b^5. b^6 is of rank less than R and hence realizes a transformation different from those realized by b^1, b^2, b^3, and b^4. But N_1 contains five other junctions realizing b^1, b^2, b^3, b^4 and b^5, and so N_1 contains at least six stroke elements. Q.E.D. The set got from $E(N)$ by replacing $b^1 \equiv Sa^1b^4$ by $b^1 \equiv Sa^1b^6$ and $b^6 \equiv Sa^3a^3$ is associated with a six-stroke-element w.f.n. that realizes all the transformations realized by N .

Because Theorem 6-2 holds it is of interest to find necessary and sufficient conditions for a deterministic net to have the property that there exists as simple a w.f.n. realizing the same transformations. We have obtained a sufficient condition for this property involving the concept of a properly driven net.

Any junction of a net N is defined to be a <u>properly driven junction</u> of N if and only if it belongs to the smallest class satisfying the following rules:

(1) each input junction and each delay output junction is properly driven, and

(2) if g and h are properly driven and $f \equiv Sgh$, then f is properly driven.

A <u>properly driven net</u> (p.d.n.) is then defined to be a w.b.n. in which every junction is properly driven. The key theorem concerning p.d.n. is formulated in terms of the following concept. A set of equations $E'(N)$ is a <u>complete subset</u> of $E(N)$ if and only if for each output junction f of N $E'(N)$ contains at least one equation of $E(N)$ of the form $f \equiv$ —— . This key theorem is:

<u>Theorem</u> 6-3: If N is p.d. $E(N)$ contains a complete subset $E'(N)$ which is unitary and regular.

<u>Proof:</u> $E'(N)$ may be constructed as follows. For each delay output junction f of N select one equation of the form $f \equiv D$ —— . Then iterate the following step: Add to $E'(N)$ any equation $f \equiv Sgh$ of $E(N)$ satisfying the following conditions: (1) there is no equation of the form $f \equiv$ —— already in

E'(N) ; (2) both g and h satisfy the condition of having a left-hand occurrence in an equation of E'(N) or being an input variable. It is obvious from its mode of construction that E'(N) is unitary and regular, and it follows from the definition of a properly driven junction (since the above procedure will catch every properly driven junction and hence every junction of the net) that E'(N) is a complete subset of E(N) . Q.E.D.

We can now prove:

<u>Theorem</u> 6-4: If N is p.d., there is a w.f.n. as simple as N realizing all the transformations realized by N .

<u>Proof</u>: The required w.f.n. is any subnet of N to which is associated a complete subset of E(N) which is both unitary and regular. Q.E.D. Theorem 6-3 also leads directly to:

<u>Theorem</u> 6-5: Every p.d.n. is deterministic.

A characteristic distinguishing p.d.n. from deterministic nets is given by:

<u>Theorem</u> 6-6: Every stroke cycle of a p.d.n. contains at least one multiple junction.

<u>Proof</u>: For if N is p.d. and contains a stroke cycle which has no multiple junctions, any complete subset of E(N) must contain the equations of this stroke cycle and hence cannot be regular. Q.E.D.

APPENDIX

In this appendix Theorem 5-8 concerning primitive recursive functions will be proved. This will be done in two steps; first a lemma concerning primitive recursive functions[‡] will be established and then the lemma will be used to prove the theorem.

A function A is said to be primitive recursive with respect to the functions $B_1, \ldots, B_I$ whenever A can be defined by the operations of primitive recursion and by substitution from the successor function, the constant functions, the identity functions, and $B_1, \ldots, B_I$. If $B_1, \ldots, B_I$ is the null sequence, A is simply said to be primitive recursive.[‡‡]

The lemma we will first establish is:

Lemma A-1: If A is definable by the schema: $A(0) = y_0, \ldots, A(M) = y_M$, where $y_0, \ldots, y_M$ are natural numbers; $A(n+M+1) = C\,[A(n), \ldots, A(n+M)]$, where C is primitive recursive with respect to $B_1, \ldots, B_I$; then A is primitive recursive with respect to $B_1, \ldots, B_I$.

Proof: Let p_y denote the y'th prime number in order of magnitude ($p_0 = 0$, $p_1 = 2, \ldots$) [p. 13]. An integer $x \geq 2$ is said to be in Gödel form whenever it is in the form $p_1^{z_1} \cdot p_2^{z_2} \cdots p_N^{z_N}$, where each z_n is a natural number and p_N is the largest prime factor of x; e.g., $2^2 \cdot 3^0 \cdot 5^1$ is the Gödel form of 20. $L(x)$ is the number of exponents (counting the zero exponents, if any) of the Gödel form of x and $nGlx$ is the n'th exponent

‡ In this appendix "function" is used in a broader sense than that defined in Section 3, since it here covers functions with any finite number of arguments and with any natural number as value.

‡‡ See Kleene, op. cit., p. 42 and "On Undecidable Propositions of Formal Mathematical Systems," notes on lectures by Kurt Gödel, Institute for Advanced Study, Princeton, New Jersey, 1934, p. 2ff.

We will assume the contents of the latter work throughout this appendix and will give page references to it in brackets. Alternative references are:

Kurt Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und Verwandter Systeme I," Monatshefte für Mathematik und Physik 38 (1931), 173-198.

S.C. Kleene, "General Recursive Functions of Natural Numbers," Mathematische Annalen 112 (1936), 727-742.

(counting from the left) of the Gödel form of x (if $x < 2$ then $L(x) = nGlx = 0$ and if $n > L(x)$ then $nGlx = 0$) [p. 13]. We next define a shifting function $Sh(x)$ which, if $L(x) > 1$ denotes the result of deleting the leftmost exponent and the rightmost prime of the Gödel form of x and shifting all exponents one place to the left:

$$Sh(x) = 2^{2Glx} \cdot 3^{3Glx} \cdots p_{L(x)-1}^{L(x)Glx} \quad ;$$

if $L(x) \leqq 1$ then $Sh(x) = 1$. E.g., $Sh(2^4 \cdot 3^9 \cdot 5^0 \cdot 7^8) = 2^9 \cdot 3^0 \cdot 5^8$, and $Sh(2^2) = 1$. The functions p, L, Gl , multiplication, and exponentiation are primitive recursive [pp. 3, 13], and hence Sh is.

We next define a function U by primitive recursion as follows:

$$U(0) = 2^{y_0} \cdot 3^{y_1} \cdots p_{M+1}^{y_M}$$

$$U(n+1) = Sh[U(n)] \cdot p_{M+1}^{C[1GlU(n), \ldots, (M+1)GlU(n)]} \quad .$$

Since C is primitive recursive with respect to the B_i's and since Sh , multiplication, and exponentiation are primitive recursive[‡], it follows that U is primitive recursive with respect to the B_i . Now the first exponent of the Gödel form of $U(n)$ is $A(n)$; i.e.,

$$A(n) = 1 \; Gl \; U(n) \quad .$$

We shall prove this by showing that

$$A(n+m) \equiv (m+1) \; Gl \; U(n) \qquad 0 \leqq m \leqq M \quad .$$

This obviously holds for $U(0)$. We shall assume it for $U(n)$ and prove it for $U(n+1)$. By the definitions of U and Sh

$$U(n+1) = 2^{2GlU(n)} \cdots p_M^{(M+1)GlU(n)} \cdot p_{M+1}^{C[1GlU(n), \ldots, (M+1)GlU(n)]}$$

and hence by the inductive assumption

$$U(n+1) = 2^{A(n+1)} \cdots p_M^{A(n+M)} \cdot p_{M+1}^{C[A(n), \ldots, A(n+M)]} \quad .$$

But by the definition of A in the lemma, the exponent of p_{M+1} equals $A(n+M+1)$, and hence

$$A(n+1+m) \equiv (m+1) \; Gl \; U(n+1) \qquad 0 \leqq m \leqq M \quad .$$

[‡]Actually it is required that these functions be primitive recursive with respect to the B_i's , but this is also the case.

Since $A(n) = 1\ \mathrm{Gl}\ U(n)$, Gl is primitive recursive, and U is primitive recursive with respect to the B_i's , A is primitive recursive with respect to the B_i's , and the lemma is proven. Q.E.D.

Consider now:

Theorem 5-8: Every transformation realized by a w.f.n. is primitive recursive.

Proof: Let N be w.f. with input junctions $a^1, \ldots, a^J$ $(J \geqq 0)$ and output junctions $b^1, \ldots, b^K$ $(K > 0)$ in regular order. If $J = 0$ every transformation realized by N is constant and periodic and the theorem is trivial, so we will assume $J > 0$.

The concept of a transformation being primitive recursive was defined as follows in Section 3: A transformation T from $a^1, \ldots, a^J$ to $T(a^1, \ldots, a^J)$ $(J \geqq 0)$ is primitive recursive whenever $T(a^1, \ldots, a^J)$ can be defined by the operations of primitive recursion and substitution from the successor function, the constant functions, the identity functions, and $a^1, \ldots, a^J$. In the terminology that has been introduced this is equivalent to requiring that $T(a^1, \ldots, a^J)$ be primitive recursive with respect to $a^1, \ldots, a^J$. The proof of the theorem will consist in showing that each b^k is primitive recursive with respect to $a^1, \ldots, a^J$.

A function W , which will correspond to the function A of Lemma A-1, which will have values $b^1_0, \ldots, b^K_0; b^1_1, \ldots, b^K_1; \ldots,$ and which is primitive recursive with respect to the a^j's , will be defined next. Let $x = y \text{ Mod } z$ be a propositional function having the value 1 when the relation holds and 0 otherwise, let $\sim x$ have the value 1 when $x = 0$ and 0 otherwise, let $x \vee y = \text{Maximum } (x,y)$, let $x \dot{-} y$ denote $x - y$ if $x \geqq y$ and otherwise 0 , and let $x{*}y$ denote the integral part of the quotient x/y ; all of these functions are primitive recursive [cf. pp. 3, 4, 14]. W is now defined:

$$W(0) = b^1_0, \ldots, W(K-1) = b^K_0; \qquad W(K) = b^1_1, \ldots, W(2K-1) = b^K_1$$

$$W(n+2K) = (n = 0 \text{ Mod } K)w_1 + \ldots + (n = [K \dot{-} 1] \text{ Mod } K)w_K \ ,$$

where the w_k are defined in the cases considered below. Note that for a given value of n all the coefficients on the right-hand side of the previous equation are 0 except one.

Case 1: b^k is a stroke output variable.
Suppose that $b^k \equiv Sb^{k_1}a^j$, where $k_1 < k$ since the b^k are in regular order. Then w_k abbreviates $\sim W(n+[2K \dot{-} k]+k_1) \vee \sim a^j_{(n+2K)*K}$. If the rightmost two variables of $b^k \equiv Sb^{k_1}a^j$ are commuted, or they are both input variables or both output variables, the definition of w_k is similar.

Case 2: b^k is a delay output variable.
If $b^k \equiv Db^{k_1}$, w_k abbreviates $W(n+[K\dot{-}k]+k_1)$; while if $b^k \equiv Da^j$, w_k abbreviates $a^j_{(n+K)*K}$.

We will prove that W is primitive recursive with respect to the a^j's by showing that it satisfies the hypothesis of Lemma A-1. Since the b^k are in regular order, each of the values $W(0), \ldots, W(M)$, where $M \equiv 2K-1$, can be determined from $E(N)$. Hence $W(0), \ldots, W(M)$ satisfy the conditions imposed by Lemma A-1 on $y_0, \ldots, y_M$. Since $x = y \text{ Mod } z$, $\dot{-}$, $*$, $+$, $\cdot$, v, and $\sim$ are all primitive recursive functions, the second line of the definition of W accords with the schema of Lemma A-1 provided that all occurrences of W on the right-hand side have arguments in the range $n, \ldots, n+2K-1$. As before, there are two cases to consider.

Case 1: If $b^k \equiv Sb^{k_1}f$ or $b^k \equiv Sfb^{k_1}$, then we have $1 \leq k$, $k_1 \leq K$, and $k_1 < k$, and hence $0 \leq [2K\dot{-}k]+k_1 \leq 2K-1$. If $b^k \equiv Sa^{j_1}a^{j_2}$, then w_k does not contain W .

Case 2: If $b^k \equiv Db^{k_1}$, then we have $1 \leq k$, $k_1 \leq K$, and hence $0 \leq [K\dot{-}k]+k_1 \leq 2K-1$. If $b^k \equiv Da^j$, w_k does not contain W .

Hence W is primitive recursive with respect to the a^j's .

We prove finally that the values of W are $b^1_0, \ldots, b^K_0; b^1_1, \ldots, b^K_1; \ldots$ by showing that

$$b^k_t = W(tK + [k \dot{-} 1]) .$$

That this holds for $tK + [k\dot{-}1] = 0, \ldots, 2K-1$ is readily verified. We assume that it holds for $tK + [k\dot{-}1] = 0, 1, \ldots, z$ where $z \geq 2K-1$ and prove that it holds for $tK + [k\dot{-}1] = z+1$ where (since $z+1 \geq 2K$) $t \geq 2$. Since $t \geq 2$

$$W(tK + [k\dot{-}1]) = W[(t\dot{-}2)K + (k\dot{-}1) + 2K]$$

and hence by the schema defining W

$$W(tK + [k\dot{-}1]) = (n = 0 \text{ Mod } K)w_1 + \ldots + (n = [K\dot{-}1] \text{ Mod } K)w_K ,$$

where $n = (t\dot{-}2)K + (k\dot{-}1)$. But then $n = (k\dot{-}1) \text{ Mod } K$ and $W(tK+[k\dot{-}1]) = w_k$. The result may now be established by proving that $w_k = b^k_t$. Two cases are considered.

Case 1: Suppose $b^k \equiv Sb^{k_1}a^j$. Then $w_k = \sim W(n+[2K\dot{-}k]+k_1) \text{ v } \sim a^j_{(n+2K)*K}$. But since $n = (t\dot{-}2)K+(k\dot{-}1)$ and $1 \leq k \leq K$, $w_k = \sim W(tK+[k_1\dot{-}1]) \text{ v } \sim a^j_t$. Since $k_1 < k$, the inductive assumption guarantees that $W(tK+[k_1 \dot{-} 1]) = b_t^{k_1}$, and hence that $w_k = \sim b_t^{k_1} \vee \sim a_t^j = b_t^k$. If the rightmost two variables of $b^k \equiv Sb^{k_1}a^j$ are commuted or they are both input variables or both output variables the proof is similar.

Case 2: Suppose $b^k = Db^{k_1}$. Then $w_k = W(n+[K \dot{-} k]+k_1)$. But since $n = (t \dot{-} 2)K+(k \dot{-} 1)$, $w_k = W([t \dot{-} 1]K+[k_1 \dot{-} 1])$. Since $t \geq 2$ and $1 \leq k$, $k_1 \leq K$, $(t \dot{-} 1)K+(k_1 \dot{-} 1) < tK+(k \dot{-} 1)$, and so by the inductive assumption we have $w_k = b_{t-1}^{k_1} = b_t^k$. Suppose $b^k = Da^j$. Then $w_k = a_{(n+K)*K}^j$. But since $n = (t \dot{-} 2)K+(k \dot{-} 1)$ and $k \leq K$, $w_k = a_{t-1}^j = b_t^k$.

Since $b_t^k = W(tK+[k \dot{-} 1])$, W is primitive recursive with respect to the a^j's , and $\cdot$, $+$, $\dot{-}$ are primitive recursive, it follows that b^k is primitive recursive with respect to the a^j's . This fact, in the light of the definition of a primitive recursive transformation, implies that the transformation realized at b^k is primitive recursive and Theorem 5-8 is proved. Q.E.D.

We conclude with a remark relating the two-stage proof given here to the function F , introduced in the text for the purpose of suggesting a method of proof. If the function A of Lemma A-1 is taken to be W , then the function U defined in the proof of the lemma has the form:

$$U(0) = 2^{b_0^1} \cdot 3^{b_0^2} \cdots p_{2K-1}^{b_1^{K-1}} \cdot p_{2K}^{b_1^K}$$

$$U(1) = 2^{b_0^2} \cdot 3^{b_0^3} \cdots p_{2K-1}^{b_1^K} \cdot p_{2K}^{b_2^1}$$

$$\vdots$$

and in fact may be shown to be the function F .

INDEX OF TERMS

The page numbers refer to occurrences of the terms where they are defined or explained.