THE UNIVERSITY OF MICHIGAN

COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS

Communication Sciences Program and Logic of Computers Group

Technical Report

CYCLES IN LOGICAL NETS

John H. Holland

UMRI Projects 2722 and 2794

under contract with:

National Science Foundation
Washington, D. C.
NSF Grant - G-4790

U.S. Army Signal Supply Agency
Laboratory Procurement Support Office
Fort Monmouth, New Jersey
Contract No. DA-36-039-sc-78057

administered by:

THE UNIVERSITY OF MICHIGAN RESEARCH INSTITUTE ANN ARBOR

August 1959

PREFACE

My curiosity concerning systems having complex patterns of feedback was first aroused during an earlier study of the action of the central nervous system (Rochester, N., J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a Cell Assembly Theory of the Action of the Brain, Using a Large Digital Computer", I. R. E. Transactions on Information Theory, IT-2,3: 80-93 (1956)). The possibility of studying feedback by means of an abstract deductive system was first suggested by a remark occuring in a paper by Arthur Burks and Hao Wang (Burks-Wang [1], p. 292 ; references are numbered in alphabetical order in Part 7 of this paper). The present paper indicates some effects of complex feedback patterns upon the behavior of an abstract class of discrete systems (logical nets). I hope that the study will suggest some properties and points for investigation of feedback in more general systems.

I would like to thank the Logic of Computers Group at The University of Michigan for providing the opportunity to carry out this research and for discussions with its other members, especially Arthur Burks and Jesse Wright, on many topics related to this paper. I would also like to thank Arthur Burks for his help in reading the first draft of this paper.

The possibility of presenting this thesis within an appropriate framework is one result of the efforts of the Committee on Communication Sciences in establishing the Communication Sciences as a discipline in their own right.

The steps in carrying the original manuscript to an approved final form were nicely handled by Ruth Lewis.

The funds for this study were supplied from grants to the Logic of Computers Group by the Signal Corps (DA 36-039 SC-78057) and the National Science Foundation (NSF-G4790).

John H. Holland

Ann Arbor

March, 1959

CONTENTS

FIGURES

1. Introduction

Feedback is a prominent structural feature of most systems which exhibit complex behavior. In fact, some systems are actually defined in terms of the role feedback is to play. It is not simple, however, to see just what conditions a given feedback pattern imposes upon a system's behavior. The problem is an example par excellence of an important class of problems arising in the investigation of automata - the class of analysis problems. The object of the usual analysis problem for automata is to derive from selected structural features of a system - feedback patterns in the present case - conditions on the system's behavior. The customary way to begin the analysis is to make a suitable abstraction of the systems involved; that is, the analysis procedure is defined over some broad class of abstract systems. If this is done properly we achieve the important result that statements about the structure logically imply statements about system behavior. Deductions about behavior will then hold true of any system (concrete or abstract) for which the structural statements hold. It is the purpose of this paper to study feedback by using the methods of logic in this way.

The study will be carried out by using the theory of logical nets to abstract the structural and behavioral properties of interest (Burks - Wright [2], Burks - Wang [1]). Thus the study will apply mainly to systems for which (at least as an approximation) time can be considered discrete and the structure as well as the number of states fixed and finite. Within this formalism structure will be represented

by the logical net diagram or the corresponding set of recursive equations. Feedback loops will be represented by cycles in the net diagram. The behavior of the logical net will be associated with properties of the state transition graph or the corresponding transition matrix. The form of this graph can in general be deduced from the recursive equations describing the structure. Thus the theory of logical nets satisfies the criterion of being able to make deductions about behavior from structural statements.

Two kinds of behavioral properties will play an important part in the development. Properties of the first kind depend upon the fact that the net-state sequence of a logical net is periodic if its input-state sequence is periodic (Burks - Wright [2], Theorem I). Properties of the second kind deal with various characteristics of state cycles in the state transition graph. Relating changes in these two kinds of properties to changes in the complexity of structural cycles will be one of the specific objectives of this paper. The overall purpose of the paper will be to produce an integrated series of definitions and operations useful for both formal and experimental investigations of the behavior of discrete systems with feedback.

2. Underlying Concepts

The theory of logical nets now has a fairly large literature and there exist several alternative formulations of its basic ideas (compare, for example, Burks - Wright [2], Kleene [4], Moore [6], Mealy [5], Burks - Wang [1], Copi- Elgot - Wright [3]). Thus most of the concepts underlying this particular study have several alternative forms. In this section, I will give a brief sketch of the form used here and then cite a paper containing the detailed development. In addition, whenever one of the underlying concepts is not in the literature, or when a new variant is required, the complete definition will be given.

The logical nets considered in this paper will be constructed of two types of elements - switching elements and delay elements. Each switching element has a fixed number, $k \geq 1$, of inputs and each delay element has exactly one input; each element has exactly one output. At any moment of time, $t = 0,1,2,\ldots$, each input and each output has just one of two states associated with it: 1 ("active") or 0 ("inactive"). For a given k-input switch let the state of the output at time t be $q(t)$. Let the state of the i^{th} input at t be $p_i(t)$. Then each of the 2^k ordered k-tuples $(p_0(t), p_1(t),\ldots,p_{k-2}(t), p_{k-1}(t))$ will be an argument for which $q(t)$ is uniquely determined. Hence $q(t)$ can be presented by means of a truth table with k argument columns and 2^k rows:

$p_0(t)$	$\cdots$	$p_{k-2}(t)$	$p_{k-1}(t)$	$q(t)$
0	$\cdots$	0	0	ϵ_0
0	$\cdots$	0	1	ϵ_1
0	$\cdots$	1	0	ϵ_2
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
1	$\cdots$	1	1	ϵ_{2^k-1}

Fig. 1 Truth Table of a k-input Switch

The output state of a delay element at time t is just the state of the input at time t - 1. This infinite class of primitive elements is essentially that discussed by Burks and Wang in part 2.3 of their paper, "The Logic of Automata", [1].

Logical nets will be constructed from the elements by identifying (connecting) the outputs of some elements with inputs of others. Thus, if an input p_i is identified with an output q_j, the state of p_i is identical at all times with the state of q_j: $p_i(t) = q_j(t)$. Element inputs not identified with element outputs in the construction process will be called net inputs. An element input which is not a net input will have its state determined at each moment by the element output with which it is identified. The state of a net input, on the other hand, is not determined by any of the net elements and must be assigned arbitrarily at each time t. The output state of each element is determined, of course, as indicated in the previous paragraph.

This paper will be concerned only with well-formed nets as defined by Copi, Elgot and Wright in part 4 of their paper, "Realization of Events by Logical Nets", [3] (or, essentially, as defined by Burks - Wright [2]). Well-formed nets can be characterized briefly as nets satisfying the following two conditions:

1) the net contains no subcycle (definition to follow) consisting only of switching elements,
2) no two distinct elements of the net have their outputs identified.

As indicated in the introduction, the purpose of this paper will be to investigate the role of cycles in logical nets. The exact definition of cycle proceeds from the concept of one element of a net

driving another. An element E_1 directly drives an element E_2, $E_1 d\ E_2$, if and only if the output of E_1 is identified with one of the inputs of E_2. A sequence of elements $F_1,\ldots,F_n$ is a drive sequence from F_1 to F_n if and only if $F_j d\ F_{j+1}$ for $j = 1,\ldots,n-1$. An element E_1 drives an element E_2, $E_1 D\ E_2$, if and only if there is a drive sequence from E_1 to E_2. Now, an element E_1 belongs to a cycle if and only if $E_1 D\ E_1$. This cycle consists of the set of all elements, E_j, such that both $E_1 D\ E_j$ and $E_j D\ E_1$. Or, more formally, a set of elements, C, in a net is a cycle if and only if, for all E_i, E_j in C, (1) $E_i D\ E_j$ and $E_j D\ E_i$ and (2) no element of the net not in C satisfies condition (1). A set of elements, C', is a subcycle if and only if, for all E_i, E_j in C', (1) $E_i D\ E_j$ and $E_j D\ E_i$ and (2) for each relation $E_i D\ E_j$ of (1) all elements of the defining drive sequence are elements of C'. It is an immediate consequence of these definitions that each element of a net belongs to at most one cycle, although it may belong to several subcycles.

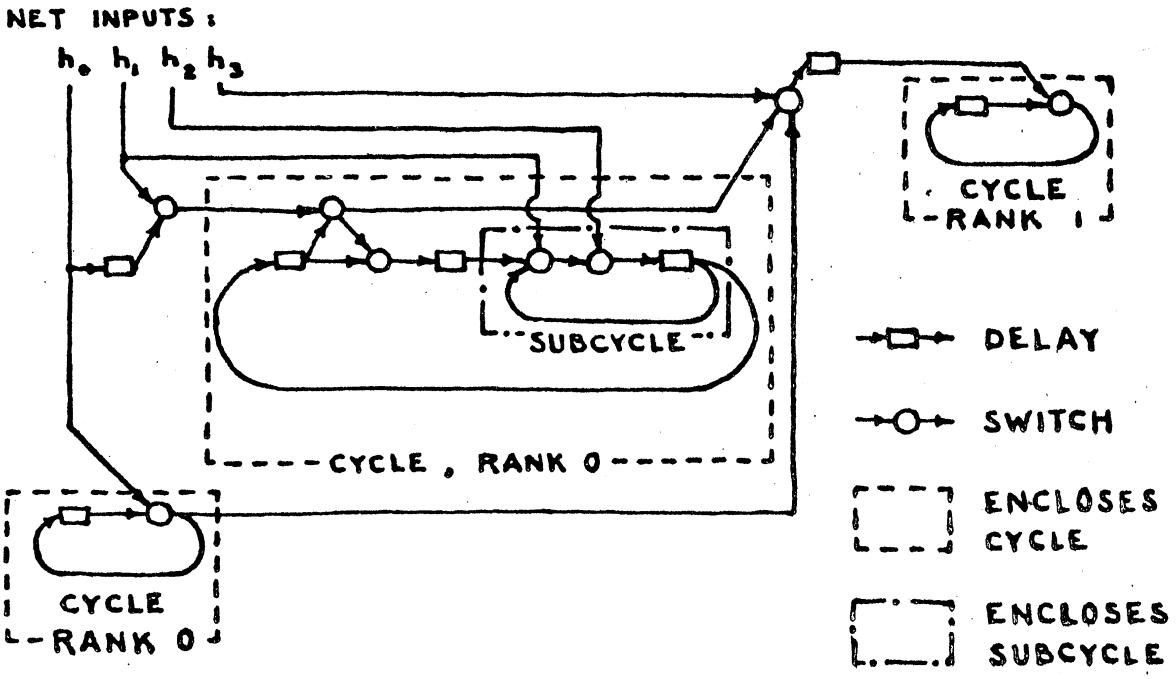


Fig. 2 A Logical Net with Cycles Marked

Note that the cycles in a well-formed net can be ranked as follows: A cycle is of rank 0 if none of its inputs is driven by an element of another cycle. A cycle is of rank r if at least one of its inputs is driven by an element of another cycle of rank r-1 and none of its inputs is driven by elements of other cycles of rank greater than r-1.

Each cycle can be reduced to a normal form which has at most one switch "between" each delay and its "predecessors". We can arrive at this normal form in the following way: A drive sequence in which all elements, other than the first and last, are switches will be called a drive sequence of switches. If there is at least one drive sequence of switches from an element E_1 to an element E_2 it will be said that E_1 drives E_2 via switches, $E_1 \underline{s} E_2$. The set of all switches belonging to drive sequences of switches from E_1 to E_2 will be called the set of switches from E_1 to E_2. Since we are dealing with well-formed nets and since the delay D_2 has but one input, there must be one and only one switch output identified with the input of D_2 when $D_1 \underline{s} D_2$ (excluding the case $D_1 \underline{d} D_2$). The set of switches from D_1 to D2 will have a total number, k, of inputs, $p_0,\ldots,p_{k-1}$, which are not identified with the output of any other switches in the set or with the output of D_1. It can easily be seen that each assignment of states to these k switch inputs uniquely determines the input state of D_2 when the output state of D_1 is given. Thus we can replace the set of switches from D_1 to D_2 by a single k + 1 input switch with k inputs identified in just the same way as the k switch inputs, $p_0,\ldots,p_{k-1}$, and the other input identified with the output of D_1. The output of this

switch is then identified with the input of D_2. Consider now the complete set of delays belonging to a given cycle. Let the delays be ordered and labelled, $D_0, D_1, \ldots, D_{n-1}$ (preferably with the relation $D_{j-1} \leq D_j$ holding for as many delays as possible). For each given j, consider all D_i in the cycle such that $D_i \leq D_j$. (There must be at least one such D_i, otherwise D_j would not be a member of the cycle.) Among all the switches belonging to one or more of the sets of switches from D_{i_1} to D_j, from D_{i_2} to $D_j, \ldots$, or from D_{i_m} to D_j, there will be a total of k switch inputs not identified with the output of any other element of these sets or with the outputs of $D_{i_1}, D_{i_2}, \ldots, D_{i_m}$. Now, in just the same way as before, a single k + m input switch can be constructed to replace the given switches. Of the inputs to this new switch, k will be identified just as the k selected inputs of the given switches, and the other m will be identified with the outputs of $D_{i_1}, \ldots, D_{i_m}$. The output of the new switch will be identified with the input of D_j. Once this is done for each D_j, the normal form of the cycle results.

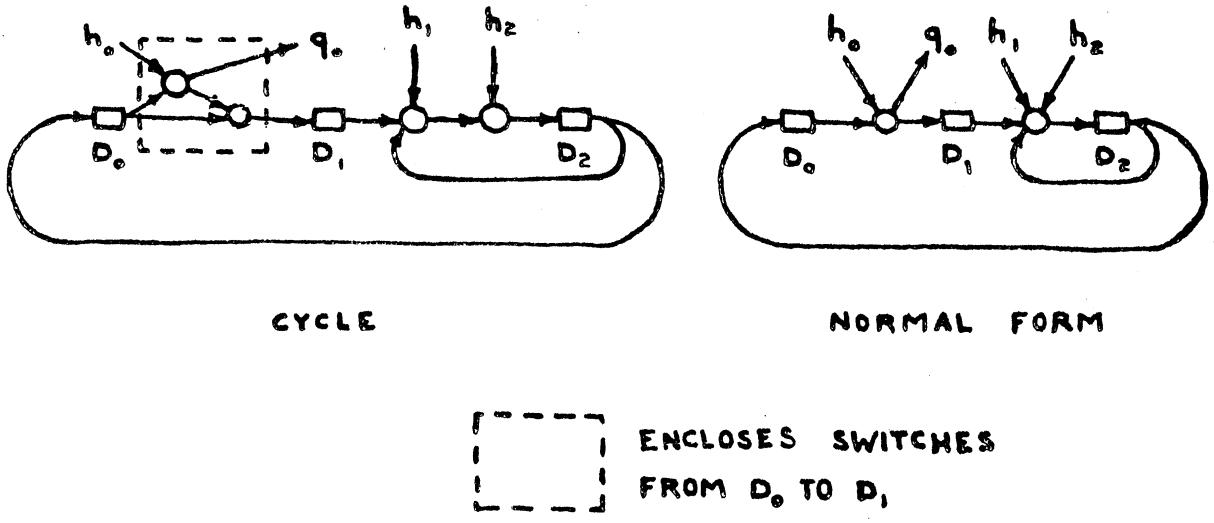


Fig. 3 A Cycle and its Normal Form

It is an immediate consequence of the construction method that the normal form of any cycle with n delays will have no more than n switches. Furthermore, each cycle input (the net inputs of a cycle when it is taken as the whole net) of the original cycle will drive via a switch exactly the same delays in the normal form as in the original form. Finally, at any time t, the input and output state of each delay in the normal form will be that of the same delay in the original form.

Note that we can always, at least formally, obtain a normal form of a cycle in which $D_{j-1 \bmod n} \leq D_{j \bmod n}$ for all j. To do this, when $D_{j-1 \bmod n} \not\leq D_{j \bmod n}$ in the given cycle, simply add an additional input to the switch constructed by the earlier method. The action of the switch is to be independent of this new input, i.e., the output of the switch is still uniquely determined by the k + m inputs initially obtained. However, formally we have a k + m + 1 input switch and can identify the output of $D_{j-1 \bmod n}$ with the newly added input. This modified normal form will simplify some of the proofs to follow.

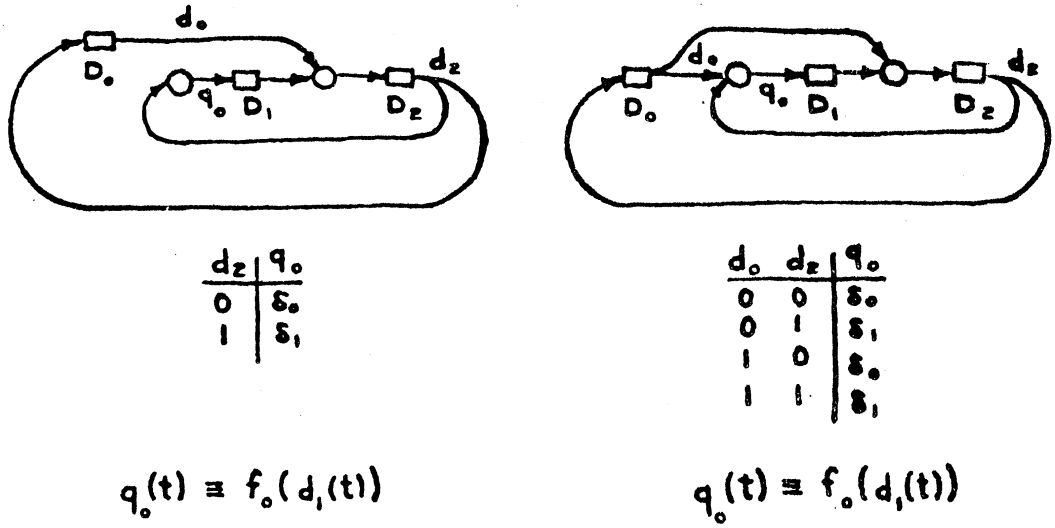


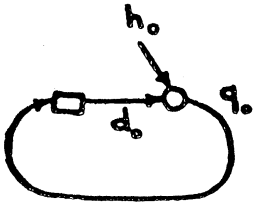
Fig. 4 A Cycle and an Equivalent Normal Form with $D_{j-1 \bmod n} \leq D_{j \bmod n}$

The definitions and discussion up to this point have been concerned with the structure of logical nets. The remainder of this section will concern the state-transition graph and behavioral properties of logical nets.

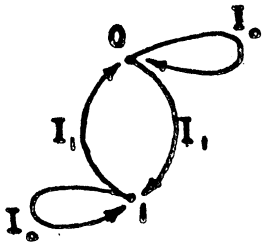
At a given time, by knowing the output state of each delay element in a net together with the state of each net input, one can determine the state of every element input and output of the net. That is, a complete description of the state of the net at any time can be obtained from knowledge of the net structure and the current states of the net inputs and delay outputs. Now, let us assume the net has $k \geq 0$ net inputs. To each of these net inputs, one of two states can be assigned at each time t; thus there are 2^k possible distinct assignments, of which one is to be chosen at each time. Any such assignment will be called an input state. If the net inputs are ordered and labelled, say $h_0,\ldots,h_{k-1}$ - and throughout the paper this will always be assumed the case - we can represent each input state by a k-tuple of ones and zeros. The delay outputs, ordered and labelled, can be treated in a similar way, the result in this case being called a net state. (See Burks - Wang [1] or Copi - Elgot - Wright [3] for detailed versions.)

The state-transition graph, as a means of picturing the behavior of a logical net, is made possible by the following fact: The net state succeeding the net state present at a time t is determined by the input state at time t. (If there are no net inputs each net state has a single invariant successor.) This transition from net state to net state under the influence of successive input states is the basic behavior of the logical net. The state-transition

graph exhibits the possible transitions and the input states (or input state successions) which accomplish them. Assuming the net has n delay outputs, there will be 2^n vertices to the graph, labelled to correspond to the 2^n net states. An edge connects vertex S_i to vertex S_j just in case one or more input states, $\{I_h\}$, cause a transition from S_i at time t to S_j at time t + 1; the edge is labelled with the set $\{I_h\}$ (cf. Moore [6]).



h_0	d_0	q_0
0	0	0
0	1	1
1	0	1
1	1	0



$I_0 \equiv (h_0 = 0)$

$I_1 \equiv (h_0 = 1)$

Fig. 5 A Cycle and its Transition Graph

Let a sequence $\{S(t)\}$ be said to have a period p if and only if, for all t greater than some positive integer a, $S(t) = S(t')$ if $t \equiv t' \bmod p$. If there is no $p' < p$ for which the preceding statement holds then the state sequence will be said to have a proper period p. A property of logical net behavior which plays a considerable role in parts of this paper can now be stated: If a net's input state sequence is periodic, then the net state sequence will also be periodic (cf. Burks - Wright [2] Theorem I). The result follows from the observation that, if the input states recur with period m and there are 2^n net states, the net state/input state combination occuring at

any time $t > m \cdot 2^n$ must repeat at $t + cm$, for some $c < 2^n$. Therefore the net state at time $t + c \cdot m+1$ must repeat the net state at $t+1$ (since each is uniquely determined by the same preceding net state/input state combination); and the input state at $t+c \cdot m+1$ must repeat the input state at $t+1$ (since both occupy the same relative position in the periodic input sequence, i.e., $t+1 \equiv t+c \cdot m+1 \bmod m$); etc. for $t+cm+2$, $t+cm+3$,... . Thus the net state sequence has a period cm. Often when studying the behavior of logical nets it will prove useful to relate the spectrum of input state proper periods to a resultant spectrum of net state periods (not necessarily proper).

3. Simple Cycles

A hint of the role of cycles in logical nets comes from the following observation: For a net with n delays and no cycles, the net state at time t is totally determined by the sequence of input states from t-n to t-1 - the net state at time t is completely independent of any net state preceding time t-n+1. Thus a net without cycles can only record the detection of an input event for at most n time-steps. In other words, if a logical net is to have "memory" or storage it must include cycles.

Upon noting the function of cycles as memory elements, one of the first questions which presents itself is: Can the full range of logical net behavior be obtained from nets using cycles of limited complexity? More precisely, in the class of well formed nets, is there a proper subset, defined in terms of some limitation on the cycles allowed, which can exhibit the full range of logical net behavior? It is fairly obvious that the number of cycles cannot be limited; this would contradict the existence of nets with arbitrarily large numbers of memory units. A possible first step would be to consider nets using only cycles with minimal feedback, i.e., cycles with no proper subcycles. A cycle of this type, which I will call a simple cycle, can be directly defined as a cycle in which each delay drives via switches exactly one other delay in the cycle. The normal form of such cycles is particularly simple.

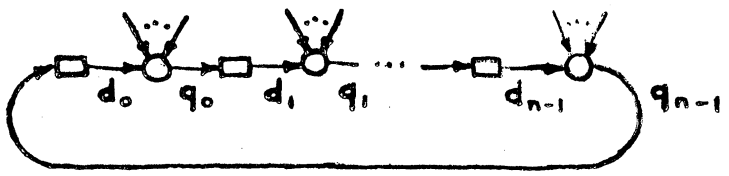


Fig. 6 Normal Form of Simple Cycle

The conjecture, then, with respect to simple cycles would be that for each logical net there is a net with, at most, simple cycles which behaves the same. The conjecture is plausible on the view that the simple cycles provide "delay line" or "reverbratory" storage of various periods while the rest of the net provides encoding, switching, decoding, and other logical operations.

Theorem 1. If the sequence of input states of a simple cycle has a proper period m and the simple cycle has n delay elements, then the sequence of net states of the cycle has a proper period 2 l.c.m.(m,n) or a divisor thereof.

Proof:

1) Let $I(t)$ be the input state to the simple cycle at time t. Let $I_j = I(j)$ be the input state at time $t = j$, $j = 0,1,\ldots,m-1$. Then, since the input states repeat with proper period m, $I(t) = I_j$ if and only if $t \equiv j \bmod m$.

2) Now reduce the simple cycle to normal form and label the delay and switch outputs as in Figure 6. Let $N_i^{t\{m\}}_{(n)}(\delta)$ be the state of the switch output labelled $q_{i \bmod n}$ when its input from the cycle (delay output $d_{i \bmod n}$) is in state δ and the net inputs are in state $I_{t \bmod m}$. If $d_{i(n)}(t)$ is the state of delay output $d_{i \bmod n}$ at time t, then for any i

$$d_{i+1(n)}(t) = q_{i(n)}(t) = N_{i(n)}^{t(m)}(d_{i(n)}(t))$$

by the delay equation and definition of $N_i^{t\{m\}}_{(n)}$.

More generally,

$$d_{i+h(n)}(t+h) = N_{i+h-1(n)}^{t+h-1(m)} N_{i+h-2(n)}^{t+h-2(m)} \cdots N_{i(n)}^{t(m)}(d_{i(n)}(t)).$$

Letting $b_0 = \text{l.c.m}(m,n)$ we have

$$d_{i+b_0(n)}(t+h_0) = d_{i(n)}(t+h_0) = N_{i+b_0-1(n)}^{t+b_0-1(m)} \cdots N_{i(n)}^{t(m)}(d_{i(n)}(t))$$

$$= N_{i-1(n)}^{t-1(m)} N_{i-2(n)}^{t-2(m)} \cdots N_{i(n)}^{t(m)} (d_{i(n)}(t))$$

since $b_0 \bmod n \equiv b_0 \bmod m \equiv 0$.

Letting $T = N_{i-1}^{t-1}\{{}_{(n)}^{(m)}\} \cdots N_{i}^{t}\{{}_{(n)}^{(m)}\}$ it follows that

$$d_{i(n)}(t+kb_0) = T(d_{i(n)}(t+(k-1)b_0)) = T \cdot T(d_{i(n)}(t+(k-2)b_0))$$
$$= T^k(d_{i(n)}(t))$$

Note that successive applications of the operator T give the states of delay output $d_{i \bmod n}$ at times t, $t+b_0$, $t+2b_0$, etc. Furthermore, if $T(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$, for some t_0, then by definition we must have $T(d_{i(n)}(t_0+b_0)) = T(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$ since $d_{i(n)}(t_0+b_0)$ $= T(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$. In other words, $T^2(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$ and, in general, $T^k(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$, once $T(d_{i(n)}(t_0)) = d_{i(n)}(t_0)$ for some t_0.

3) For any given i and t the form of the T operator is fixed and can be given by a table of the following form:

δ	T
0	T(0)
1	T(1)

where $T(\delta) = \begin{cases} 0 \\ 1 \end{cases}$.

4) If T(0) = 0, T(1) = 1 then we must have $T(d_{i(n)}(t)) = d_{i(n)}(t)$ whatever the state of delay output $d_{i \bmod n}$ at time t. Hence, $T^k(d_{i(n)}(t)) = d_{i(n)}(t)$ and the state of $d_{i \bmod n}$ at time t must repeat at least every b_0 time-steps thereafter.

5) If T(0) = 0, T(1) = 0 then $T(d_{i(n)}(t)) = 0$, although for $t_0 < b_0$ $d_{i(n)}(t_0)$ may equal 1. However, $T^2(d_{i(n)}(t_0)) = T(0) = 0$. Hence, $T(\delta_{i(n)}(t_0+b_0)) = \delta_{i(n)}(t_0+b_0)$ and, again, the state of $d_{i \bmod n}$, after at most b_0 time-steps repeats with period b_0.

6) A similar argument holds for T(0) = 1, T(1) = 1.

7) Finally, if $T(0) = 1$, $T(1) = 0$, we have $T^2(d_{i(n)}(t)) = d_{i(n)}(t)$ and in general $T^{2k}(d_{i(n)}(t)) = d_{i(n)}(t)$. Hence, the state $d_{i(n)}(t)$ repeats with period $2b_0$ in this case.

8) The full argument obviously holds for any t and any i. Thus the state sequence of each delay output in the cycle and hence the net state sequence repeats every $2b_0$ units of time. If $2b_0$ is divisible by b, it is possible that $d_{i(n)}(t) = d_{i(n)}(t+jb)$ for all j such that $jb \leq b_0$ and for all i. In such case, the proper period of the net state sequence would be b, a division of $2b_0$. Thus $2b_0 = 2$l.c.m.(m,n) or a divisor thereof is the net state period of the cycle, irrespective of the switching elements used.

9) Since in T we find for each pair (i,j), $\begin{matrix} i = 0,1,\ldots,n-1 \\ j = 0,1,\ldots,m-1 \end{matrix}$, exactly one occurence of N_i^j, it is easy to show (by giving the rule for constructing the corresponding truth tables) that there in fact exist switching elements which will yield proper net state period $2b_0$.

In nets constructed with more than one simple cycle there can be subnets having no cycles "between" the simple cycles. The following definitions are intended to give a precise interpretation of this statement. A drive sequence from an element F_1 to an element F_2 will be called an n.c.-drive sequence if none of the elements in the sequence other than the first and last, belongs to a cycle. A set of elements, A, n.c.-drives a set of elements, B, if there is at least one n.c.-drive sequence from some element of A to some element of B. The net consisting of all elements, other than elements of A and B, belonging to n.c.-drive sequences from A to B is the n.c.-net from A to B.

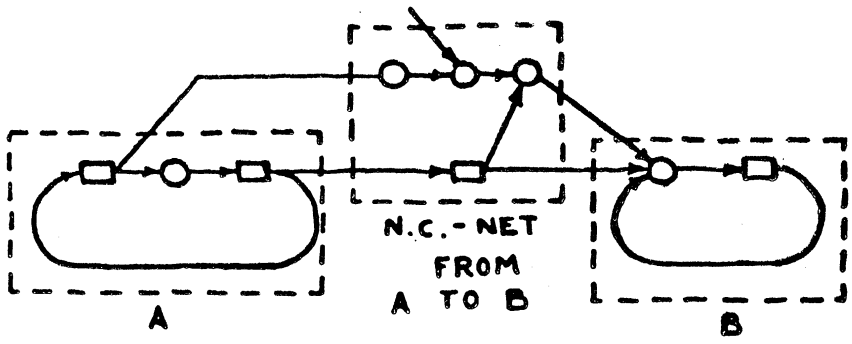


Fig. 7 An N.C.-Net

Theorem 2. If the sequence of input states to a net without cycles (e.g., an n.c.-net) has a proper period m, then the net state sequence of the n.c.-net has a proper period m or a divisor thereof.

Proof:

1) In order to simplify the proof let each element in the net without cycles by given a rank as follows: (1.1) if all the inputs of the element are net inputs of the given net, its rank is zero, (1.2) if an input of the element is identified with the output of an element of rank r-1, and no input is identified with the output of an element of rank higher than r-1, then the element is of rank r. Since an n.c.-net contains no cycles each element has a unique rank.

2) Now, the states of the inputs of each element of rank zero must repeat with period m since these states are merely components of the net input state. Therefore, since the output state of a switch is uniquely determined by the set of input states, the output state of each switch of rank zero repeats every m units of time. The switch may, of course, repeat its output state more frequently; e.g., the switch output state may be 1 for any input argument, in which case. its output state would repeat with period p = 1. Thus the output state

sequence of a switch of rank zero has a period m or a divisor thereof. It follows directly from the delay equation that the output state of a delay of rank zero repeats every m units of time.

3) If the output state of each element of rank $r' < r$ repeats with period m, then, by the same observations as in the case of rank zero, the output state of an element of rank r repeats with period m. Thus, by induction, the output state of any element of a net without cycles repeats every m units of time. Hence, the net state sequence of a net without cycles has a proper period m or a divisor thereof.

We can now proceed to the general case with respect to the conjecture mentioned at the outset of this section; that is, the case of nets in which all the cycles are simple cycles.

Theorem 3. A net in which all of the cycles are simple, having $n_1,\ldots,n_k$ delay elements, respectively, with a sequence of input states of proper period m, will have a net state sequence of period 2^{r_0+1} l.c.m.$(m,n_1,\ldots,n_k)$, where r_0 is the maximum of the cycle ranks.

Proof:

1) To begin with,note that every element of a net, N, is driven by one or more of the net inputs of N, except those elements belonging to a cycle having no cycle inputs (i.e., the cycle, considered as a net, has no net inputs).

2) Let A_0 be the set of elements satisfying the following two conditions: (2.1) at least one input of the element is a net input of N, (2.2) the element does not belong to a cycle. Let C be the set of elements belonging to a given cycle of rank 0 (see Part 2). Any element driving an element of C must be driven by an element of A_0 because C is of rank 0 and hence no element belonging to a cycle can

drive an element of C. Let P be the net consisting of the n.c.-net from A_0 to C together with the elements of A_0. The net includes all elements of N which drive elements of C. Since the net inputs of P are just the net inputs of N, the net state sequence of P must be of period m by Theorem 2. Thus the input state sequence of the cycle inputs to C must be of period m. Since C is a simple cycle the results of Theorem 1 apply - the net state sequence of C has a proper period 2 l.c.m.(m,n), or a divisor thereof, where n is the number of delays belonging to C.

3) Now, define inductively a set of nets N_j, for $j = -1,0,1,\ldots,r_0$, where r_0 is the maximal rank of the cycles in the net N. The net N_j consists of the following elements with their inputs identified as in N:

3.1) all elements of N_{j-1} (where N_{-1} is the set A_0),

3.2) all elements belonging to n.c.-nets from N_{j-1} to cycles of rank j,

3.3) all elements belonging to cycles of rank j.

4) Consider the net N_0. If the k_0 simple cycles in N_0 have $n_0,\ldots,n_{k_0-1}$ delays, respectively, then as shown above the i^{th} cycle will have a net state sequence of period 2 l.c.m.(m,n_i), $i = 0,\ldots,k_0-1$. Each associated P net will have a net state period m. It follows directly that the proper net state period of N_0 will be a divisor of

$$\text{l.c.m.}(m,\ 2\ \text{l.c.m.}(m,n_0),\ldots,\ 2\ \text{l.c.m.}(m,n_{k_0-1}))$$
$$= 2\ \text{l.c.m.}(m,n_0,\ldots,n_{k_0-1})$$

5) Let $n_0,\ldots,n_{k_0-1},n_{k_0},\ldots,n_{k_1-1},\ldots,n_{k_j-1}$ be the number of delays belonging, in order, to each of the simple cycles from rank 0 through rank j.

6) Assume the net N_{j-1} has a net state sequence of period $p = 2^j$ l.c.m.$(m,n_0,\ldots,n_{k_{j-1}-1})$. By substituting N_{j-1} for A_0 and p for m in step (2) we see that a cycle C_i of rank j must have a net state sequence of period

$$2 \text{ l.c.m.}(p,n_i) = 2 \text{ l.c.m.}(2^j \text{l.c.m.}(m,n_0,\ldots,n_{k_{j-1}-1}), n_i)$$
$$= 2^{j+1} \text{l.c.m.}(m,n_0,\ldots,n_{k_{j-1}-1},n_i)$$

Applying the reasoning of step (4) we see then that the proper net state period of N_j will be a divisor of

$$\text{l.c.m.}(m,2^{j+1} \text{l.c.m.}(m,n_0,\ldots,n_{k_{j-1}})$$

7) Thus by induction on j, the net N, when the input states repeat with period m, will have a proper net state period which is a divisor of

$$P = 2^{r_0+1} \text{l.c.m.}(m,n_0,\ldots,n_k).$$

Again, for reasons similar to those noted at the end of Theorem 1, there exist combinations of switching elements and cycles giving N a net state sequence of proper period p.

Corollary Let it be required that a net be in a chosen net state S_0 if and only if the number of occurences, p, of a distinguished input state I_0 satisfies the equation $p \equiv 0 \bmod j$. (Simply, the net is required to "count", modulo j, the occurences of input state I_0.) For nets in which all cycles are simple, j must equal 2^b for some positive integer b. (Such nets can only "count" modulo a power of 2.)

Proof:

Let the distinguished input state repeat with a proper period m. Then, since the net is to be in a unique net state S_0 for each j occurences of the distinguished input state, it must have a net state which repeats with proper period $p = jm$ for all m. Or, by Theorem 3,

$$p = jm = 2^{r_0+1} \text{l.c.m.}(m,n_0,\ldots,n_k).$$

This equation can only hold for all m if $j = 2^{r_0+1}$, since if m is chosen equal to $k(\text{l.c.m.}.(n_0,\ldots,n_k))$ the above equation reduces to

$$jm = 2^{r_0+1}m.$$

If this result is not surprising, it at least shows the over-simplification present in the idea that the main function of cycles in a system is to provide "memory" or storage of information. Here we have systems with any number of cycles of arbitrary lengths (arbitrary recycling times) which have a very limited range of behavior, not because we restrict the complexity of the switching elements used, but because the cycles are limited in the complexity of their feedback patterns.

4. Input-Independent Cycles

The results of Part 3 show the behavior of a logical net to be severely restricted if the complexity of the net cycles is sufficiently limited. Moreover, the limitation on complexity need not concern the number of cycles or the number of delays in a cycle, but only the number of feedback loops (subcycles) per cycle. The effect of increasing the number of feedback loops in a cycle thus becomes a salient point of the study of cycles in logical nets.

In the present section, I will start out by relating properties of the state-transition graph to changes in feedback in a class of input-independent cycles called locally-balanced cycles. Just as cycles are important features of net structure, so cycles in the transition graph are important to net behavior. To distinguish the two kinds of cycles, I will speak of net cycles and state cycles, respectively. The relation between locally-balanced cycles and the resulting state cycles will be a key to the behavior of more general net cycles.

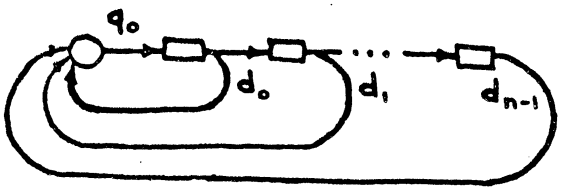
Locally-balanced cycles can be defined in the following way: In the truth table corresponding to a switching element let rows $2j$ and $2j+1$, for $j = 0,1,\ldots,2^{n-1}-2$, be called simply the j^{th} pair. Let the function values determined by the two arguments of the j^{th} pair be ϵ_{2j} and ϵ_{2j+1} respectively. A switching element will be called locally-balanced if $\epsilon_{2j} = \bar{\epsilon}_{2j+1}$ for all pairs, $j = 0,1,\ldots, 2^{n-1}-2$. A locally balanced cycle satisfies the following conditions:

1) One switching element occurs in the cycle and that element is locally-balanced.

2) There are $n > 0$ delay elements in the cycle. The output of

the switch is identified with the input of delay d_0. The output of delay d_j is identified with the input of delay d_{j+1}, $j = 0,\ldots,n-2$.

3) The switch has n inputs. The j^{th} input of the switch (the j^{th} column of the associated truth table) is identified with the output of delay d_j, $j = 0,\ldots, n-1$.



d_0	d_1	...	d_{n-2}	d_{n-1}	q_0
0	0	...	0	0	ϵ_0
0	0	...	0	1	ϵ_1
0	0	...	1	0	ϵ_2
0	0	...	1	1	ϵ_3
...	...		...		...
1	1	...	1	0	ϵ_{2n-2}
1	1	...	1	1	ϵ_{2n-1}

$$\epsilon_{2j} = \bar{\epsilon}_{2j+1}$$

Fig. 8 A Locally-Balanced Cycle
and the Truth Table of Its Switch

Theorem 4. The state-transition graph of any locally-balanced cycle consists only of disjoint cycles of states.

Proof:

1) Consider, at any given time t, the ordered n-tuple $(p_0(t),\ldots,p_{n-1}(t))$ of the states of the n inputs to the switch in a locally-balanced cycle. By definition of a locally-balanced cycle, this n-tuple is identified with the ordered n-tuple of delay output states at time t, $(d_0(t),\ldots,d_{n-1}(t))$. Thus each of the 2^n net states of the cycle is represented by the argument part of a line of the switching element truth table.

2) If the net state of the cycle at t is given by the j^{th} line of the truth table, then the net state of the cycle at t+1 is simply given by the ordered n-tuple with ϵ_j as its first digit and the value

of $p_i(t)$ as its $i+1^{th}$ digit. That is:

$$(d_0(t+1),\ldots,d_{n-1}(t+1)) = (\epsilon_j, p_0(t),\ldots,p_{n-2}(t))$$

where the argument of line j has in effect been "shifted one to the right", ϵ_j being "shifted in", $p_{n-1}(t)$ being "shifted out". The truth table, thus extended, becomes derived transition table for the locally-balanced cycle:

s(t)				s(t+1)			
$d_0(t)=p_0(t)$	...	$d_{n-2}(t)=p_{n-2}(t)$	$d_{n-1}(t)=p_{n-1}(t)$	$d_0(t+1)=q(t)$	$d_1(t+1)=p_0(t)$	...	$d_{n-1}(t+1)=p_{n-2}(t)$
0	...	0	0	ϵ_0	0	...	0
0	...	0	1	ϵ_1	0	...	0
0	...	1	0	ϵ_2	0	...	1
0	...	1	1	ϵ_3	0	...	1
	...					...	
1	...	1	0	ϵ_{2^n-2}	1	...	1
1	...	1	1	ϵ_{2^n-1}	1	...	1

where s(t) is the net state of the cycle at time t.

Fig. 9 The Derived Transition Table of a Locally-Balanced Cycle

3) Now, the j^{th} pair of the locally-balanced switch gives rise to a pair of successor n-tuples, $s_{2j}(t+1) = (\epsilon_{2j}, p_0(t),\ldots,p_{n-2}(t))$ and $s_{2j+1}(t+1) = (\epsilon_{2j+1}, p_0(t),\ldots,p_{n-2}(t))$. s_{2j} and s_{2j+1} have identical digits in the last n-1 places by step (2); furthermore no other pair of successors has the same ordered set of digits in the last n-1 places. The first digit of s_{2j} is the binary complement of the first digit of s_{2j+1} since $\epsilon_{2j} = \bar{\epsilon}_{2j+1}$ by definition of a locally-balanced switch. Therefore the argument states of the j^{th} pair map into distinct n-tuples, s_{2j} and s_{2j+1}, which occur nowhere else on the right of the derived transition table. In other words, the derived

transition table is a 1 - 1 mapping of the 2^n net states onto themselves. Such a mapping is a permutation on the net states and, by elementary group theory, permuations can always be reduced to a product of disjoint permutation cycles. These permutation cycles correspond directly to disjoint state cycles of the transition graph.

Note that the state-transition graph of an input-independent cycle consists only of disjoint state cycles just in case the cycle is backwards deterministic in the sense of Burks - Wang [1; p.286]. Using this fact, and noting that $\epsilon_{2_j} = \epsilon_{2_{j+1}}$ implies $s_{2_j} = s_{2_{j+1}}$, we can restate Theorem 4 in a stronger form:

Theorem 4'. Let C be an n-delay cycle with one (arbitrarily chosen) n-input switch which is connected just like the switch in a locally-balanced cycle. C will be backwards deterministic if and only if the switch is locally-balanced.

In what follows, a pair will be said to be normally oriented if $\epsilon_{2_j} = 0$, $\epsilon_{2_{j+1}} = 1$. A pair will be said to be inversely oriented if $\epsilon_{2_j} = 1$, $\epsilon_{2_{j+1}} = 0$. The simplest locally-balanced cycles result when the pairs associated with the switch are either all normally oriented or all inversely oriented. When this is the case the output of the switch, $q(t)$, is independent of all argument columns except the last, $p_{n-1}(t)$. Thus in effect the cycle is an input-independent simple cycle. The next theorem gives some properties of the state-transition graphs of these simplest locally-balanced cycles.

Theorem 5. Let L be a locally-balanced cycle with n delays. If all the pairs of the switch are normally oriented, a state-cycle with exactly p states occurs if and only if $p = 1$ or, for $p > 1$, g.c.d.$(p,n) = p$. There will be two state-cycles with $p = 1$ and, for $p > 1$, there will be

$n_p = \frac{2^p - 2^{p'}}{p}$ state-cycles, where p' is the next number smaller than p which is the length of a state-cycle. If all the pairs of the switch are inversely oriented, a state-cycle with exactly p states occurs if and only if g.c.d.(p,2n) = p and p does not divide n. The number of state-cycles having p elements is again given by n_p.

Proof:

1) The sequence of states in a state-cycle C of an n delay locally-balanced cycle can be represented by a binary sequence of the following form:

1.1) the sequence is doubly infinite,

1.2) any set of n consecutive digits from the sequence $\delta_j, \delta_{j+1}, \ldots, \delta_{j+n-1}$ represents a state s_j of the state-cycle C,

1.3) given a set of n consecutive digits, $\delta_j, \delta_{j+1}, \delta_{j+n-1}$, which represent state s_j, the set of digits $\delta_{j+1}, \delta_{j+2}, \ldots, \delta_{j+n}$ represents the successor, s_j', of s_j in the state-cycle C.

That such a sequence exists follows from the form of the transition table as derived in Theorem 4 (the right-hand entry of a line in the table being related to the left-hand entry just as in condition (1.3) above, but with order reversed).

2) Consider first the case where the switching element in the locally-balanced cycle has all of its pairs normally-oriented. Then in the binary sequence associated with any one of the disjoint state-cycles we have $\delta_j = \delta_{j+n}$ because, in the derived transition table for the normally oriented case, $p_{n-1}(t) = q(t)$ (see Theorem 4, step (2)). Furthermore, if u is a multiple of the number of elements in the state-cycle then $\delta_j = \delta_{j+u}$. Finally, if u is the length of a state-cycle (and not a multiple thereof) then $\delta_{j_0} \neq \delta_{j_0+u_0}$ for each $u_0 \mid u$ and in each case, a $j_0 < u-u_0$.

3) Combining the three requirements of step (2) we have

3.1) $\delta_{j0} = \overline{\delta}_{j0+u} = \overline{\delta}_{j0+u0+ju}$, $j = 0,1,\ldots$

3.2) $\delta_{j0} = \delta_{j0+n} = \delta_{j0+j'n}$, $j' = 0,1,\ldots$

These equations can be satisfied if and only if,

3.3) $u_0+ju \neq j'n$, for any $u_0 \mid u$ and any j , j'

or g.c.d.$(\frac{u}{u_0},\frac{n}{u_0}) \neq 1$ for any $u_0 \mid u$

4) Consider the alternatives (4.1) g.c.d.(n,u) = u and (4.2) g.c.d.(n,u) = u' < u:

4.1) g.c.d.(n,u) = u $\Leftrightarrow$ n = bu for some integer b, whence g.c.d.$\left(\frac{u}{u_0}, \frac{n}{u_0}\right)$ = g.c.d. $\left(\frac{u}{u_0},\frac{bu}{u_0}\right) = \frac{u}{u_0}$ for any $u_0 \mid u$; therefore g.c.d.(n,u) = u $\Rightarrow$ u is a state-cycle length (and not a multiple thereof).

4.2) g.c.d.(n,u) = u' < u $\Rightarrow$ u' $\mid$ u $\Rightarrow$ for $u_0 = u'$, g.c.d.$\left(\frac{u}{u_0} , \frac{n}{u_0}\right) = 1$ and therefore the equations (3.1) and 3.2) are satisfied by u' $\mid$ u, whence u is a multiple of the state-cycle length.

5) Thus, for a locally-balanced cycle with n delay elements and a normally-oriented switch, p is the length of a state-cycle if and only if g.c.d.(n,p) = p. That is p must be a factor of n.

6) In order to determine the number, n_p, of state-cycles, of length p we note that there are 2^p binary sequences of length p. However, $2^{p'}$ of these sequences will have been assigned to cycles of length p' or divisors thereof, where p' is the number next smaller than p such that g.c.d.(p',n) = p' or, in other words, p' is the next smaller state-cycle length. Therefore there are $2^p - 2^{p'}$ binary sequences which correspond to state-cycles of length p. However, for each state-cycle of length p there are p binary sequences which represent the cycle

(obtained by shifting any one representative j places to the right, for $0 \leq j < p$). Thus $n_p = \frac{2^p - 2^{p'}}{p}$.

7) For the case with all the pairs of the switching element inversely oriented the proof procedure is much the same except that $\delta_j = \bar{\delta}_{j+n}$ in this case. Thus equation (3.3) becomes

$$u_0 + ju \neq n + j'2n, \text{ for any } u_0 \mid u \text{ and any } j, j'.$$

This equation is satisfied if, and only if

$$g.c.d.(u, 2n) = u \text{ and } u \nmid n$$

or $u = 2^{n_1+1} n'$ where $n = 2^{n_1} \cdot n_f$, $n' \mid n_f$, and n_1 is the highest power of 2 in the prime factorization of n. The number of state-cycles of length p is again given by $n_p = \frac{2^p - 2^{p'}}{p}$ (the same reasoning as before applying in this case - p' being the next smaller state-cycle length w.r.t.p).

The state-cycles of the locally-balanced cycle with a normally-oriented switch, which I will call normal state-cycles, figure basically in the present study. Part of the reason for this lies in the following operation: an inversion consists in changing a given pair of a locally-balanced switch from normally-oriented to inversely-oriented or vice versa. The result of an inversion is a new locally-balanced switch produced from the given one. It follows directly from the definition of a locally-balanced switch that any locally-balanced switch can be transformed into any other by a succession of inversions. Thus, for example, any locally-balanced switch can be produced by using a succession of inversions on a normally-oriented switch. The next five theorems will explore the relations between normal state-cycles, inversions, and the transition graphs of locally-balanced cycles.

In the proofs and at other points from here on the state represented by a given binary n-tuple will, where convenient, be labelled by the

decimal equivalent of the corresponding binary number. Thus $(0,0,\ldots,0)$ becomes 0, $(0,\ldots,0,1,0)$ becomes 2, and $(1,1,\ldots,1)$ becomes 2^n-1.

Theorem 6. Let E_1 be the switching element of a locally-balanced cycle; let E_2 be the switching element derived from E_1 by an inversion on the j^{th} pair, i.e. on $(\epsilon_{2j},\epsilon_{2j+1})$; and let s_{2j} and s_{2j+1} be the arguments of the j^{th} pair. If s_{2j} and s_{2j+1} belong to different state-cycles, C_1 and C_2, in the transition graph w.r.t. E_1, then the transition graph w.r.t. E_2 will be the same as that for E_1 except that C_1 and C_2 will be united into a single state-cycle consisting exactly of all of the states belonging to C_1 and C_2. If s_{2j} and s_{2j+1} belong to the same state-cycle, C, in the transition graph w.r.t. E_1, then the transition graph w.r.t. E_2 will be the same as that for E_1 except that C will be separated into two disjoint state-cycles which together include all of the states belonging to C.

Proof:

The transition table for a locally-balanced cycle, as derived from the switching element's truth table, is unchanged by an inversion except for the lines corresponding to the inverted pair. Let s_{2j} and s_{2j+1} be the left-hand entries of these two lines and s'_{2j}, s'_{2j+1} their respective successors (right-hand entries) before the inversion. After the inversion the successor of s_{2j} will be s'_{2j+1} and the successor of s_{2j+1} will be s'_{2j}. From Theorem 4 one of two cases must hold for s_{2j} and s_{2j+1}, either they belong to different state-cycles or else they belong to the same state-cycle.

Case 1) s_{2j}, s_{2j+1} belong to different state-cycles C_1, C_2.

After the inversion the succession from s'_{2j} to s_{2j} within C_1 is unchanged, thus each element of C_1 appears in turn (since there were

no elements of C_1 between s_{2j} and s'_{2j} all are present in this succession). However, the successor of s_{2j} is s'_{2j+1} which belongs to C_2. The succession from s'_{2j+1} to s_{2j+1} is undisturbed and every element of C_2 appears in this succession. Finally the successor of s_{2j+1} is s'_{2j} which completes the new cycle (since we began the succession with s'_{2j}). All elements of C_1 and C_2 belong to the resulting state-cycle.

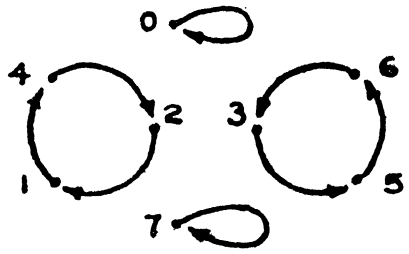
INVERSIONS

NONE

DERIVED TRANSITION TABLE

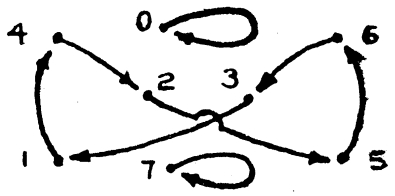
	d_0	d_1	d_2	q_0	d_1	d_2
PAIR 0	0	0	0	$\epsilon_0=0$	0	0
	0	0	1	$\epsilon_1=1$	0	0
PAIR 1	0	1	0	$\epsilon_2=0$	0	1
	0	1	1	$\epsilon_3=1$	0	1
PAIR 2	1	0	0	$\epsilon_4=0$	1	0
	1	0	1	$\epsilon_5=1$	1	0
PAIR 3	1	1	0	$\epsilon_6=0$	1	1
	1	1	1	$\epsilon_7=1$	1	1

TRANSITION GRAPH



PAIR 1 INVERTED

AFTER INVERSION OF PAIR 1 DERIVED TRANSITION TABLE IS THE SAME EXCEPT $\epsilon_2 = 1$ AND $\epsilon_3 = 0$.



PAIRS 1 AND 2 INVERTED

AFTER INVERSION OF PAIRS 1 AND 2 TRANSITION TABLE IS THE SAME EXCEPT $\epsilon_2 = 1$, $\epsilon_3 = 0$, $\epsilon_4 = 1$, AND $\epsilon_5 = 0$.

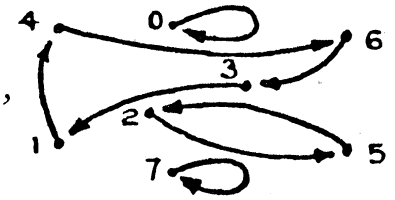


Fig. 10 Effects of Inversions on the Transition Graph of a Locally-Balanced Cycle

Case 2) s_{2j}, s_{2j+1} belong to the same state-cycle C.

Let the segment of C from s_{2j} through s_{2j+1} be D_1 and the other segment from s_{2j+1} through s_{2j} be D_2. The first element of D_1 is

s'_{2j} and the last s_{2j+1}. After inversion the succession in D_1 is unchanged but s'_{2j} becomes the successor of s_{2j+1}. Thus D_1 becomes a state-cycle. Similarly D_2 becomes another, disjoint, state-cycle. The effect of the inversion has been to "pinch" the original cycle in two at s_{2j}, s_{2j+1}.

Theorem 7. In any transition graph containing normal cycles, there are at most two inversions which can, individually, connect a given pair of normal cycles (C_1, C_2). There are no inversions which can separate a given normal cycle into two state-cycles.

Proof:

1) Let $\ldots \delta_j, \delta_{j+1}, \ldots, \delta_{j+n-1}, \delta_{j+n}, \ldots$ be the binary sequence associated with the state-cycle C_1 of the hypothesis and $\ldots \delta'_j, \delta'_{j+1}, \ldots, \delta'_{j+n-1}, \ldots$ the sequence associated with C_2 (see step (1) of Theorem 5). Since C_1 and C_2 are normal cycles they will each be of length n or a divisor thereof by Theorem 5. Consequently in the above sequences $\delta_j = \delta_{j \bmod n}$ and $\delta'_j = \delta'_{j \bmod n}$.

2) If there is an inversion connecting C_1 and C_2 then there is a mapping $V(\delta_j) = V(\delta_{j \bmod n}) = \delta'_{(j+h_1) \bmod n}$ such that $\delta_j = V(\delta_j)$, except for some one $k_1 < n$ and j_1 such that $j_1 \bmod n \equiv k_1$ for which $\delta_{j_1} = \delta_{k_1} = \overline{V(\delta_{k_1})} = \overline{\delta'_{k_1+b_1}}$. That is, for an inversion to connect C_1 and C_2 we must have the state $(\delta_k, \delta_{k-1}, \ldots, \delta_0, \delta_n, \ldots, \delta_{k+1})$ of C_1 belonging to the same pair as the state $(V(\delta_k), \ldots, V(\delta_{k+1}))$ of C_2 (see case (1) of Theorem 6). This means that $V(\delta_j) = \delta_j$ except $V(\delta_k) = \overline{\delta}_k$ (see the last sentence of step (1) of Theorem 5).

3) If there is a second inversion connecting the two state-cycles then there is a second mapping $V'(\delta_j) = \delta'_{(j+b_2) \bmod n}$, with

$b_2 \neq b_1$ and $V'(\delta_j) = \delta_j$ except $V'(\delta_{k_2}) = \overline{\delta}_{k_2}$ for some one $k_2 < n$ and $k_2 \neq k_1$.

4) Let β = g.c.d. $(|b_2-b_1|, n)$ and form equivalence classes $\{\delta_{(j\beta + u) \bmod n}\}$ for $0 \leq u < \beta$ and $j = 0,1,2,\ldots$. For u such that neither k_1 nor k_2 belongs to the equivalence class selected by u we must have all elements of the class equal. This can be seen by noting that $\delta_u = \delta'_{(u+b_1) \bmod n} = \delta'_{(u+b_2) \bmod n}$, under V and V', and that $\delta_{(u+b_2-b_1) \bmod n} = \delta'_{(u+b_2) \bmod n}$ under V'; whence it follows $\delta_u = \delta_{(u+b_2-b_1) \bmod n}$ and, by repeating the procedure, $\delta_u = \delta_{(u+ r(b_2-b_1)) \bmod n}$ for any r, which can be reduced to the form $\delta_u = \delta_{(u+j\beta) \bmod n}$. Now consider the equivalence class which includes δ_{k_1}. By the reasoning of the preceding line and the fact that $\delta_{k_1} = \overline{\delta}'_{k_1+b_1}$ we have $\delta_{k_1} = \delta_{(k_1+(b_2-b_1)) \bmod n}$ and $\delta_{(k_1+r(b_2-b_1)) \bmod n} = \delta_{(k_1+(r+1)(b_2-b_1)) \bmod n}$, for all r such that $r(b_2-b_1) \not\equiv 0 \bmod n$, unless $(k_1+r(b_2-b_1)) \bmod n = k_2$ for some r. That is, unless k_1 and k_2 belong to the same equivalence class, we obtain $\delta_{k_1} = \overline{\delta}_{(k_1-(b_2-b_1)) \bmod n} = \overline{\delta}_{k_1 \bmod n}$ which is a contradiction. Thus k_1 and k_2 belong to the same equivalence class. The above relations can be used directly to construct binary sequences satisfying them. Therefore, some state-cycles satisfying the conditions of the hypothesis can be connected by either of two inversions.

5) If there were a third inversion connecting the two state-cycles, then $V''(\delta_j) = \delta'_{(j+b_3) \bmod n}$, $b_3 \neq b_1$, $b_3 \neq b_2$ and $V''(\delta_j) = \delta_j$ except $V''(\delta_{k_3}) = \overline{\delta}_{k_3}$ for some $k_3 \neq k_1, k_2$. Now k_3 must belong to one of the equivalence classes above. We can assume, without loss of generality, that $k_1 < k_2 < k_3 < n$. If k_3 belongs to $\{\delta_{(j\beta+u) \bmod n}\}$ which does not contain k_1 and k_2 we obtain $\delta_u = \overline{\delta}_u$, a contradiction.

But if k_3 belongs to the class containing k_1 and k_2 we have

$$(k_1 + r_1(b_2-b_1)) \bmod n = k_2$$

$$(k_2 + r_2(b_2-b_1)) \bmod n = k_3$$

$$(k_3 + r_3(b_2-b_1)) \bmod n = k_1$$

whence $\delta_{k_1} = \overline{\delta}_{k_2} = \delta_{k_3} = \overline{\delta}_{k_1}$ which is again a contradiction. Therefore a third inversion connecting the two cycles is impossible, and hence, there can be at most two inversions able to connect the two cycles.

6) The second part of the theorem follows immediately from the observation that all the states of a given normal cycle have exactly the same number of digits equal to 1 in their binary representations. Since the second argument of any pair must, by definition, always have one more digit equal to 1 than the first argument, the two arguments of a pair can never belong to the same normal cycle. Therefore, because the conditions of case (2) of Theorem 6 are not satisfied, no single inversion can separate a normal cycle into two state-cycles.

Theorem 8. There is a locally-balanced cycle having a transition graph in which all 2^n net states belong to a single state-cycle; i.e., the net will have a net state sequence of period 2^n.

Proof:

1) By Theorem 4 we know that the transition graph of any locally-balanced cycle consists only of disjoint cycles of states. For the purposes of this proof, a state-cycle C will be said to be composed only of pairs of states if one argument of a pair of the switch occurs as a state in C just in case both arguments of the pair occur in C. In order to prove the present theorem we shall first show that any state-cycle composed only of pairs of states must include the states (0) and (1).

Assume there is at least one state-cycle composed only of pairs. Let $(2j, 2j+1)$ be one of the pairs occuring in this state-cycle. By the definition of a locally-balanced cycle (see the derived transition table of Figure 9) either $(2j)$ or $(2j+1)$ must have an element of the pair $(2[\frac{j}{2}], 2[\frac{j}{2}] + 1)$ as a successor state, where $[\frac{j}{2}]$ is the integral part of $\frac{j}{2}$. Since the state-cycle is supposed to be composed only of pairs, both arguments of the pair $(2[\frac{j}{2}], 2[\frac{j}{2}] + 1)$ must belong to the state-cycle. Continuing this line of reasoning we see the states $([\frac{j}{2}]), ([\frac{j}{2^2}]),\ldots,(1), (0)$ must be elements of the state-cycle.

2) From the preceding paragraph it follows that the transition graph of any given locally-balanced cycle contains at most one state-cycle composed only of pairs of states. That is, any state-cycle composed of pairs must contain (0) and (1), but this pair can belong to at most one state-cycle.

3) Consider now the n delay locally-balanced cycle, L_0, having a switching element with all of its pairs normally oriented. Invert the pair having arguments (0,1). By Theorem 5 the elements of the first pair belong to two different state-cycles. Thus by Theorem 4 the result of the inversion will be to join these two state-cycles to form a single new state-cycle, C_1. The transition graph of the locally-balanced cycle, L_1, obtained by the inversion will thus have one less state-cycle.

4) If the new state-cycle, C_1, is not composed entirely of pairs, then there must be a smallest pair $(2j, 2j+1)$ such that one of its elements belongs to a state-cycle disjoint from C_1 in the transition graph for L_1. But then an inversion on this pair will yield a new state-cycle C_2 and a locally-balanced cycle, L_2, which has one less

state-cycle than L_1. The process can be continued until one first reaches a state-cycle, C_j, formed for L_j, which is composed only of pairs.

5) But C_j will be composed only of pairs just when C_j contains all 2^n states of L_j. Otherwise there would be state-cycles of L_j disjoint from C_j. These state-cycles would not be composed wholly of pairs (see step (2)). Therefore, we could apply the same process of successive inversion that was originally applied to C_1, to pairs belonging to these state-cycles (pairs not belonging to C_j). C_j will be undisturbed since none of these pairs contains any of its elements. Furthermore the process cannot terminate at any stage in another state-cycle composed only of pairs, since this would give a locally-balanced cycle with two state-cycles composed only of pairs. But then the process must continue until all elements of the cycles disjoint from C_j are joined in a cycle C'_k. But in that case C'_k must consist only of pairs - the pairs not belonging to C_j. This again contradicts step (2); hence there can be no state-cycles disjoint from C_j. C_j, as constructed, contains all 2^n states of the locally-balanced cycle L_j.

The next theorem begins a direct investigation of the effect of increasing the number of feedback loops in a cycle. The theorem basically concerns input-independent locally-balanced cycles in which some of the feedback loops to the switch have been omitted, i.e., cycles in which the switch receives $k \leq n$ inputs from the cycle.

Just before Theorem 5 I mentioned that, in a locally-balanced cycle, use of a switch with all pairs normally oriented or all pairs inversely oriented, in effect, converts the cycle to a simple cycle. This observation can now be generalized: A switching element will

be said to be of order k if and only if there are k numbers, $0 \leq i_0 < \ldots$ $\ldots < i_{k-1} \leq n-1$, such that all arguments with the same values for $p_{i_0},\ldots,p_{i_{k-1}}$ determine the same output value, q(t), for the switch. If this is true for k and for no $k_1 < k$ the switch will be said to be properly of order k. The output state, q(t) of a switch properly of order k depends only on the state of inputs $p_{i_0},\ldots,p_{i_{k-1}}$; thus in a cycle the switch could be replaced by a switch with $k \leq n$ inputs identified with delay outputs $d_{i_0},\ldots,d_{i_{k-1}}$

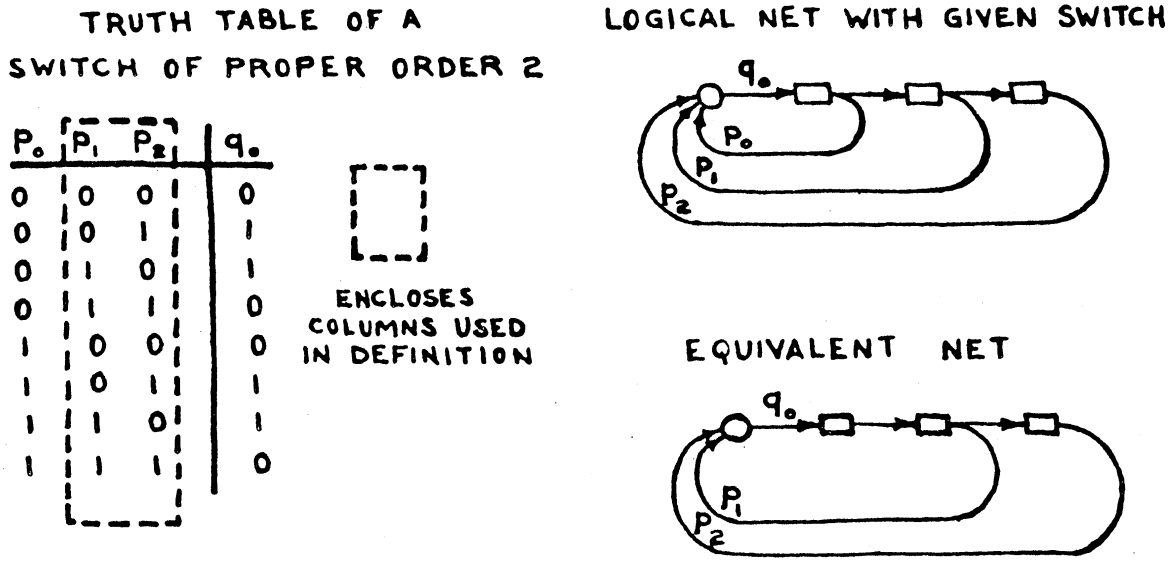


Fig. 11 A Switch Properly of Order 2

For a locally-balanced switch, $i_{k-1} = n-1$. This is the case because, in each pair determined by giving values to $p_0,\ldots,p_{n-2}$, $q(t) = \epsilon_{2j}$ when $p_{n-1} = 0$ and $q(t) = \epsilon_{2j+1} = \bar{\epsilon}_{2j}$ when $p_{n-1} = 1$. That is, different values of p_{n-1} give rise to different values of q(t). Using these facts the definition of order can be recast for locally-balanced switches in terms of orientation of pairs: A locally-balanced switch is of order k if and only if there are k-1 numbers,

$0 \leq i_0 < \ldots < i_{k-2} < n-1$, such that all pairs with the same values for $p_{i_0},\ldots,p_{i_{k-2}}$ have the same orientation.

The proof of Theorem 9 will describe some of the changes occuring in the transition graph of a locally-balanced cycle as the order of its switch is increased (i.e., as the number of feedback in the cycle is increased). The statement of this theorem as well as that of some succeeding theorems can be considerably shortened by making the following definition: The state-cycle partition of a net N with n delays is a partition of the set of 2^n net-states satisfying the following conditions:

1) one subset of the partition consists just of those states which do not belong to a state-cycle in the transition graph of N.

2) the remaining net-states are separated into subsets such that two states belong to the same subset if and only if they belong to the same state-cycle in the transition graph of N.

Theorem 9. The set of state-cycle partitions associated with the set of locally-balanced cycles having n delays and a switching element of order k properly includes the set of state-cycle partitions associated with any collection of locally-balanced cycles having n delays and switching elements properly of order $k' < k$.

Proof:

1) We begin by noticing again that all of the states in a given normal state-cycle have the same number of ones in their binary representation. Thus the set of normal state-cycles associated with a locally-balanced cycle can be classified according to the number of ones in the binary representations of their states.

2) For a locally-balanced switching element to be properly of order $1 < j \leq n$ there must be at least one set of inversely oriented pairs among the sets of pairs specified by the definition of a j^{th} order switching element (i.e., among the sets of pairs defined by the sets of 2^{n-j+1} binary n-tuples, $(p_0,\ldots,p_{n-2})$, which can be formed for each fixed assignment of values to the j-1 components $p_{i_0},\ldots,p_{i_{j-2}}$).

3) Consider a switching element properly of order j which has exactly one set of inverted pairs and let the set be the one obtained by setting $p_{i_0} = p_{i_1} = \ldots = p_{i_{j-2}} = 0$. As we range over the arguments of these inverted pairs, the number of digits equal to 1 in any given argument will range from zero to n-j+1. For this switch no inverted pair will have an argument with more than n-j+1 components equal to 1. Thus a locally-balanced cycle, L, using the given switch, will have a transition graph in which each state-cycle having a state with more than n-j+1 ones in its binary representation will be a normal state-cycle. Furthermore, for the given switch, there will be one and only one inverted pair whose 2^{nd} argument has n-j+1 components equal to 1. For this reason the state-graph of L will have one state-cycle consisting of a normal cycle of n-j+1 ones connected to a normal cycle of n-j ones (which in general will be connected with other normal cycles). Thus one of the normal state-cycles with n-j+1 ones in the binary representation of its states will not be present as a state-cycle among the set of disjoint state-cycles of L (rather it will be part of a larger state-cycle).

4) If we consider an assignment of values to $p_{i_0},\ldots,p_{i_{j-2}}$ other than $p_{i_0} = \ldots = p_{i_{j-2}} = 0$, then some normal state-cycles of more than n-j+1 ones will be connected to another normal cycle. Hence the switch

specified in step(3) is the only switch properly of order j which will yield a locally-balanced cycle for which all state-cycles having states with more than n-j+1 ones in their binary representation are normal while some of the other state-cycles are not normal.

5) For switches of order $0 < j' < j$ some normal cycle of $n-j'+1 > n-j+1$ ones will be connected to another normal cycle. For switches of order 0 all state-cycles are either normal or else all are non-normal. Therefore there is a switch or order j which, in a locally-balanced cycle, produces transition graph different from any produced by switches of order $j' < j$. (Actually this can be generalized: Let T_h represent the subset of the partition of normal state-cycles which contains all normal state-cycles of h ones. Then only switches of order j or greater will yield state-graphs, for any choice of $c < j$, where all cycles other than those in T_c, $T_{c+1},\ldots,T_{c+n-j+1}$ are normal while some cycles in T_c, $T_{c+1},\ldots,T_{c+n-j+1}$ are connected by inversions.)

6) Since the set of switching elements of order j includes all switches of order $j' < j$, the set of state-graphs produced by switches of order j in a locally-balanced cycle properly includes the set of state-graphs produced by switches of order $j' < j$.

Let a pair be called <u>unbalanced</u> if $\epsilon_{2j} = \epsilon_{2j+1}$. Starting from the normally-oriented n-input switch any n-input switch can be produced by unbalancing selected pairs after carrying out a properly chosen set of inversions. Using this fact Theorem 10 and its corollary extend the results of Theorem 9 to every input-independent cycle with one switch.

<u>Theorem 10</u>. The set of state-cycles for a given input-independent cycle with one switch, E_1, of order k is a subset of the set of state-cycles of an associated locally-balanced cycle with a switch, E_2, of order $k' \geq k$.

No locally-balanced cycle with a switch of order less than k' includes, as a subset, the set of state-cycles of the cycle with switch E_1.

Proof:

1) Consider the derived transition table (cf. Theorem 4, step(1)) of a net cycle with a single switch. Just as in the case of a locally-balanced cycle, all digits of the successor state in each row, excepting the first digit ϵ_j, are determined independently of the switch used. Only ϵ_j is determined by the switch output.

2) Now consider the unbalanced pairs, if any, of the switch E_1. In an unbalanced pair both arguments, s_0 and s_1, have the same successor state, s', in the derived transition table. Thus one of the two arguments, say s_0, cannot belong to a state-cycle; otherwise some other state in the state-cycle would have to have two successors in the derived transition table, an impossibility for input-independent cycles. The other of the two arguments, s_1, may or may not belong to a state-cycle. If s_1 belongs to a state-cycle orient the corresponding pair in the locally-balanced switch, E_2, so that s_1 has the same successor s'. (E.g., if s_1 is the first argument of the pair and the first digit of s' is $\epsilon_{2_j} = 1$, then the pair in the locally-balanced switch E_2 should be inversely oriented.) If s_1 does not belong to a cycle, the corresponding pair in E_2 can have either orientation and still satisfy the first part of the theorem; however, in step (5) we will see that sometimes one of the two orientations will have to be specified in order to satisfy the second part of the theorem.

3) For each balanced pair in E_1 the corresponding pair in E_2 should be balanced with the same orientation.

4) The switch E_2 is fully specified once the instruction of step (2) or step (3), whichever applies, has been carried out with respect to

each pair of E_1. Each argument of E_1 which belongs to a state-cycle also belongs to a state-cycle of the locally-balanced cycle with switch E_2 and in both cases the successor state is the same. Thus the set of state-cycles with respect to E_1 must be a subset of the set of state-cycles with respect to E_2.

5) Consider the j^{th} pair of E_1. Let $p_{i_0},\ldots,p_{i_b}$ be the smallest set of digits in the first argument of the j^{th} pair such that all other pairs with the same values for these digits in their first argument also have the same values for $(\epsilon_{2j}, \epsilon_{2j+1})$. If this j^{th} pair is balanced then so must be the other pairs with the same values for $p_{i_0},\ldots p_{i_b}$ and they must have the same orientation. All corresponding pairs in E_2 by definition will have the same orientation. In this case the values of $(\epsilon_{2j}, \epsilon_{2j+1})$ for E_2 will also be determined by the digits $p_{i_0},\ldots,p_{i_b}$. If the j^{th} pair of E_1 is unbalanced then so must be the other pairs specified by $p_{i_0},\ldots,p_{i_b}$. If all of these pairs either have the same argument, e.g., the first, belonging to a cycle or else neither argument belonging to a cycle then all of the corresponding pairs in E_2 can be given one orientation. Then, as before, all pairs in E_2 with same values for digits $p_{i_0},\ldots,p_{i_b}$ will have the same orientation and hence the same values for $(\epsilon_{2j}, \epsilon_{2j+1})$. Finally, some of the given unbalanced pairs in E_1 may have their first argument belonging to a cycle and some the second. Then the corresponding pairs in E_2 will not have the same orientation (see step (2)). Thus for E_2 additional digits $p_{i_{b+1}},\ldots,p_{i_{b+a}}$ will be necessary in this last case to specify just those pairs with the same orientation. Thus we see that E_2 will in general have a proper order $k' > k$. Furthermore it follows directly from the preceding argument that no locally-balanced switch can have proper order less than that of E_2 and still satisfy

step (4).

Corollary The set of state-cycle partitions associated with the set of input-independent net-cycles each having n delays and one switch of order k properly includes the set of state-cycle partitions associated with any collection of input-independent net-cycles each having n delays and one switch properly of order $k' < k$.

Proof:

1) Consider a switch properly of order k, with at least one unbalanced pair, inserted in an input-independent cycle with one switch. By the first two lines of step (2) in Theorem 10, the transition graph of such a cycle will contain at least one state which does not belong to a state-cycle. Thus the transition graph of step (2) of Theorem 9, since it consists just of state-cycles, cannot occur for any switch, balanced or unbalanced, of proper order $k' < k$.

2) Since the set of switching elements of order k includes all switches of order $k' < k$, the corollary follows.

Let a net-cycle of order k be defined as a net-cycle whose normal form contains at least one switch of order k and contains no switch properly of order $k' > k$. Using this definition Theorem 11 extends the results of the last two theorems to all input-independent cycles.

Theorem 11. The set of state-cycle partitions of the set of all n-delay net cycles of order k properly includes the set of state-cycle partitions of any collection of n'-delay, $n' \leq n$, net-cycles of order $k' < k$.

Proof:

1) Note first that, with respect to net-cycles having n delays, there is only one normal state-cycle containing a state with n ones in its

binary representation. Furthermore this normal state-cycle has but one element, namely (1,1,...,1).

2) Consider a switch E_1 which, when used in an input independent net-cycle with one switch, causes the state (1,1,...,1) to have a successor state with fewer than n digits equal to 1. (Since the net-cycle has but one switch, the successor will clearly have to be (0,1,...,1).) Thus in the transition graph of the net-cycle using E_1 no normal state-cycle with n ones will appear.

3) Now consider the set of all n-delay net-cycles, in normal form (see Part 2), which have E_1 as one of the switches. The normal state-cycle with n ones will not be present in the transition graph of any of these net-cycles - this can be deduced as follows: The state (1,1,...,1) is the only element of the normal state-cycle with n ones; therefore the successor of (1,1,...,1) will have to be (1,1,...,1) if the normal state-cycle with n ones is to be present in a transition graph. When the net-state, s(t), is (1,1,...,1) the output state of each delay in the net-cycle is $d_j(t) = 1$, $j = 0,\ldots,n-1$. If the output of E_1 is connected to the input of delay d_1, then $d_1(t+1) = 0$ since the output state of E_1 is $q_0(t) = 0$ when the argument is (1,1,...,1). The output state of a switch whose output is identified with the input of delay d_j, $j = 0,2,3,\ldots,n-1$, can be either $q_{j-1}(t) = 0$ or $q_{j-1}(t) = 1$. If $q_{j-1}(t) = 0$ then $d_j(t+1) = 0$; if $q_{j-1}(t) = 1$ then $d_j(t+1) = 1$. The net-state $s(t+1) = (d_0(t+1), d_1(t+1) = 0,\ldots, d_{n-1}(t+1))$ can at best have one less 1 than the net-state $(d_0(t) = 1, d_1(t) = 1,\ldots,d_{n-1}(t) = 1)$. Therefore no n-delay net-cycle including in its normal form a switch such as E_1 will have the normal state-cycle with n ones present in its transition graph.

4) Let E_k be the switch of proper order k which gives the transition graph described in step (3) of Theorem 9. All state-cycles having states with more than n-k+1 ones in their binary representation will be present in this transition graph as normal state-cycles. However, at least one of the normal state-cycles with n-k+1 ones, C_k , will not be present in the transition graph. That is, one state s_{2h+1} normally belonging to C_k will have as a successor s'_{2h+1}, a state not belonging to C_k. By the construction of step (3) in Theorem 9, s'_{2h+1} will have n-k ones, positions $d_1, d_{i_0+1}, \ldots d_{i_{k-2}+1}$ being equal to zero, $i_{k-2} \leq n-1$. For convenience I will assume $i_0 = 1$, $i_1 = 2, \ldots, i_{k-2} = k-1$. The proof can be generalized by inserting d_{i_0} for d_1, etc. throughout steps (5) -(11).

5) Now consider the set of all n-delay net-cycles, in normal form, with E_k as one of the switches and in which no switch is of order greater than k. The next six steps of this proof will show that no net-cycle in this set has a transition graph with the following property: The transition graph consists only of state-cycles and each state-cycle having an element with more than n-k ones in its binary representation is a permutation of a normal state-cycle.

6) Assume that a transition graph with the property of step (5) is possible. Let the output, q_0 of E_k be identified with the input of delay d_1. Then, in particular, the state s_{2h+1} of step (4) must have a successor state with n-k+1 ones. At time t, for the net-state to be $s_{2h+1}(t)$, we have $d_1(t) = 0, \ldots, d_{k-1}(t) = 0$ and $d_0(t) = 1$ along with $d_k(t) = 1, \ldots, d_{n-1}(t) = 1$. Since $q_0(t) = 0$ we know that $d_1(t+1) = 0$. Thus $s'_{2h+1}(t+1)$ can contain n-k+1 digits equal to 1 only if $d_{j_0+1}(t+1) = 1$ for some $2 \leq j_0 \leq k$. But this means that switch q_{j_0} must produce an output state $q_{j_0}(t) = 1$; furthermore this switch must be of proper order $\leq k$ by step (5). Two possibilities present themselves:

6.1) The switch q_{j_0} has an output state $q_{j_0}(t) = d_{j_0}(t)$ for all arguments with more than n-k+1 digits equal to 1 in the associated truth table.

6.2) The switch q_{j_0} has an output state $q_{j_0}(t) = \overline{d_{j_0}(t)}$ for some argument with more than n-k+1 digits equal to 1 in the associated truth table.

7) Consider first a switch of type (6.1). The next three steps of the proof will show that a switch of type (6.1) requires the presence, in the net cycle, of an additional switch q_{r_0} which is of type (6.2). That is, q_{r_0} is required if the net-cycle is to have a transition graph with the property given in step (5):

8) For the argument $s_{2h+1}(t) =$

$(d_0(t) = 1, d_1(t) = 0, d_2(t) = 0, \ldots, d_{k-1}(t) = 0, d_k(t) = 1, \ldots, d_{n-1}(t) = 1)$

the switch q_{j_0} must have an output state $q_{j_0}(t) = d_{j_0}(t) = 1$ by the first part of step (6). Note, however, that the switch q_{j_0} must be of proper order $k_0 \leq k$ by step (5). That is, the output state $q_{j_0}(t)$ must be the same for all arguments having the same values for a given set, 0_{k_0}, of k_0 components of the argument (see the discussion of proper order preceding Theorem 9). There are now two alternatives for 0_{k_0}:

8.1) The components $d_1, \ldots, d_{k-1}$ are all included in 0_{k_0}.

8.2) At least one of the components $d_1, \ldots, d_{k-1}$ is not included in 0_{k_0}.

9) If (8.1) is the case then we look at a net-state $s(t)$ chosen so that $d_1(t) = 0, \ldots, d_{k-1}(t) = 0$ and all other components are 1 except, say, $d_0(t) = 0$. $s(t)$ has a binary representation with n-k digits equal to 1. The successor state $s'(t)$ will have at least n-k+1 digits equal to 1 because of the action of q_{j_0} <u>unless</u> there is a third switch q_{j_1}

in the normal form of the net-cycle. In other words, unless there is a third switch q_{j_1}, one of the normal state-cycles with n-k+1 ones will not be present in the transition graph. But if there is a switch q_{j_1} we must start again from step (6) and determine whether q_{j_1} is of type (8.1), type (8.2) or type (6.2). If q_{j_1} is in turn of type (8.1) we will require a switch q_{j_2} and so on until:

9.1) the allowed number of switches, n, is exceeded, in which case the contention of step (5) is proved,

or 9.2) a switch q_j is of type (6.2) or type (8.2); if type (6.2) the statement of step (7) is proved; if type (8.2) we apply step (10).

10) If (8.2) is the case, let d_u, $1 \leq u \leq k-1$, be the component not belonging to 0_{k_0}. Since d_u does not belong to 0_{k_0} we can set $d_u(t) = 1$ and still have $q_{j0}(t) = 1$. But then, unless $u = j_0$, $q_{j0}(t) = 1 = \overline{d_{j_0}(t)}$ for an argument with n-k+2 digits equal to 1, contradicting (6.1). However, if $u = j_0$ the two arguments

$$s_{2h+1}(t) = (d_0(t) = 1,\ d_1(t) = 0,\ldots,d_{j_0}(t) = 0,\ldots,d_{k-1}(t) = 0,\ d_k(t) = 1,\ldots,d_{n-1}(t) = 1)$$

and $$s_{2v+1}(t) = (d_0(t) = 1,\ d_1(t) = 0,\ldots,d_{j_0}(t) = 1,\ldots,d_{k-1}(t) = 0,\ d_k(t) = 1,\ldots,d_{n-1}(t) = 1)$$

have the same successor state

$$s'_{2h+1}(t) = (d_0(t) = 1,\ d_1(t) = 0,\ldots,d_{j_0+1}(t) = 1,\ldots,\ d_{k-1}(t) = 0,\ d_k(t) = 1,\ldots,d_{n-1}(t) = 1).$$

Thus a third switch q_{r_0} is required if the transition graph is to be composed just of state-cycles. If q_{r_0} is chosen to change the successor state of s_{2h+1} the problem returns to step (6) and, as in step (9), either the allowed number of switches is eventually exceeded or else we must at some point introduce a switch of type (6.2). If q_{r_0} is chosen

to change the successor state of s_{2v+1} then q_{r_0} is automatically a switch of type (6.2), s_{2v+1} having n-k+2 digits equal to 1.

11) From steps (7) - (10) we see that, if we are to satisfy the hypothesis of step (6), a switch of type (6.2) must occur in the net-cycle. However, a switch of type (6.2) requires that a state s_{2w+1} with $n-k_1 > n-k+1$ digits equal to 1 have a successor with a different number of digits equal to 1. Thus yet another switch, q_{j_1}, will be required if the normal state-cycle containing s_{2w+1} is to be present in the state-transition graph. But now we can apply to q_{j_1} the reasoning from step (6) onward. Ultimately a set of switches q_{j_0}, $q_{j_1},\ldots,q_{j_m}$ will result. The last, q_{j_m} will perforce affect an argument with n digits equal to 1. But there is only one argument with n ones, (1,1,...,1), and by step (3) no additional switches can correct for the case in which (1,1,...,1) has a successor state with fewer digits equal to 1. Thus we see that all cases lead to a contradiction of the assumption of step (6). Hence the assertion of step (5) follows.

12) The theorem for $n' = n$ and $k' < k$ follows from the assertion of step (5) by virtue of steps (4) and (5) of Theorem 9 and step (1) of the Corollary to Theorem 10. To extend the theorem to $n' \leq n$ we note that in an n-delay cycle, N_c, the switches $q_{i'}$, for $i' = n'-1, n', \ldots,n-2$, can be chosen so that $q_{i'}(t) = 0$ for all arguments. Then by choosing $q_{n-1},q_0,q_1,\ldots,q_{n'-2}$ so that they are independent of argument columns $p_{n'}, p_{n'+1},\ldots,p_{n-1}$ we have, in effect, a cycle N'_c with n' delays (together with a string of n-n' delays whose output states are always zero). The first n' digits of any net-state of N_c will be exactly the digits of the corresponding net-state of N'_c. Thus there is a state-cycle partition of an n-delay net-cycle of order k which can be mapped directly on the state-cycle partition of an arbitrary n'-delay,

$n' \leq n$, net-cycle of order $k' < k$. The theorem for $n' \leq n$ and $k' < k$ follows.

Theorem 11 provides a basis for several comments on the effect of increasing the number of feedback loops in a logical net. Here I will make just one comment, reserving others for the more general context of Part 5: Consider the case of an experimenter presented with a black box (cf. Moore's gedanken experiments [6]) about which he is given the following information:

1) all elements in the box belong to a single input-independent net-cycle,
2) the box has one output for each delay element inside,
3) at any time the box can be set to an arbitrary "initial" net-state and observed for as long as desired.

Theorem 11 tells us that there is a net with at most k feedback loops through each switch which will make the black box behave in a fashion impossible for any net with $n' \leq n$ delays and $k' < k$ feedback loops through each switch. Moreover, the set of behaviors possible for such black boxes when each switch receives at most k feedbacks properly includes the set of behaviors possible for boxes having $n' \leq n$ delays and $k' < k$ feedbacks to each switch.

5. Cycles in General

The object of this section will be to relate the behavior of net-cycles in general to the behavior of input-independent net-cycles. The basis of this relation is the nature of the truth table of a switching element in an arbitrary net-cycle:

Let N_c be an n-delay net-cycle in normal form having a total of k distinct switch inputs identified with elements not belonging to N_c, i.e., net cycle inputs. The truth table of each switching element in N_c can be regarded as having a normal form with k argument columns $h_0,\ldots,h_{k-1}$ corresponding to the k net-cycle inputs and n argument columns $p_0,\ldots,p_{n-1}$ corresponding to the n net-cycle delay outputs. If a given switch in N_c has $j < n$ inputs identified with delay outputs $d_{i_0},\ldots,d_{i_{j-1}}$ of N_c then the truth table output q(t) will of course depend only upon the j columns $p_{i_0},\ldots,p_{i_{j-1}}$ of the n columns $p_1,\ldots,p_{n-1}$. That is, with respect to the n columns $p_1,\ldots,p_{i_{j-1}}$, the normal form of the truth table will be of order j. Similarly if the given switch has $b < k$ net-cycle inputs then q(t) will depend only upon b columns $h_{v_0},\ldots,h_{v_{b-1}}$ of the k columns $h_0,\ldots,h_{k-1}$. Each switch in the net-cycle, regardless of the number of its inputs, can thus be given a standard truth table with k+n argument columns and one output column.

Note that the k+n argument columns of the truth table of each switch in N_c are identical when they are given the order $h_0,\ldots,h_{k-1}$, $p_0,\ldots,p_{n-1}$. In the normal form of the net-cycle N_c, each delay in the cycle, d_i, has its input identified with the output of one of the switches, q_i, in the cycle so that $d_i(t+1) = q_i(t)$. Furthermore $p_i(t) = d_i(t)$. If the n switch output columns are arranged in the order $q_{n-1},q_0,q_1,\ldots,q_{n-2}$ at the left of the k+n argument columns the

result is a transition table where the transition from $(d_0(t),\ldots,d_{n-1}(t))$ to $(d_0(t+1),\ldots d_{n-1}(t+1))$ depends upon the value of the net-cycle input state $(h_0(t),\ldots,h_{k-1}(t))$.

I(t)			s(t)			s(t+1)	
$h_0(t)$	...	$h_{k-1}(t)$	$d_0(t) = p_0(t)$	...	$d_{n-1}(t) = p_{n-1}(t)$	$d_0(t+1) = q_{n-1}(t)$	... $d_{n-1}(t+1) = q_{n-2}(t)$
0	...	0	0	...	0	$\epsilon_{n-1,0}$	... $\epsilon_{n-2,0}$
0	...	0	0	...	1	$\epsilon_{n-1,1}$	... $\epsilon_{n-2,1}$
...	...	...	....	...	...	...	... ...
0	...	0	1	...	1	$\epsilon_{n-1,2^n-1}$	... $\epsilon_{n-2,2^n-1}$
0	...	1	0	...	0	$\epsilon_{n-1,2^n}$	... $\epsilon_{n-2,2^n}$
...	...	...	...	...	...	...	... ...
1	...	1	0	...	0	$\epsilon_{n-1,2^{k+n-1}}$	... $\epsilon_{n-2,2^{k+n-1}}$
...	...	...	...	...	...	...	... ...
1	...	1	1	...	1	$\epsilon_{n-1,2^{k+n}-1}$	... $\epsilon_{n-2,2^{k+n}-1}$

Fig. 12 Derived Transition Table
for a General Cycle with Input

Now fix a value, I_0, for $I(t) = (h_0(t),\ldots,h_{k-1}(t))$ and consider, in the derived transition table for N_c, the 2^n rows having the given values for the arguments $h_0(t),\ldots,h_{k-1}(t)$. The 2^n rows so selected constitute the transition table of an n-delay net-cycle N_{I_0} which has n switches in normal form, each with n inputs. Of necessity, N_{I_0} is an input-independent cycle in which the switch at position q_i has, for arguments $p_0(t),\ldots,p_{n-1}(t)$, an output $q_i(t) = \epsilon_{i,x\cdot 2^n+y}$, where x = the decimal equivalent of the binary number $h_0(t)\cdot 2^{k-1}+,\ldots+h_{k-1}(t)\cdot 2^0$ and y = the decimal equivalent of $p_0(t)\cdot 2^{n-1}+\ldots+p_{n-1}(t)\cdot 2^0$. That is, at each position in N_{I_0} we have a switch whose output for an argument

$(p_0(t),\ldots,p_{n-1}(t))$ is simply the value $q_i(t)$ given for the corresponding n argument values of $p_1,\ldots,p_n$ in one of the selected 2^n rows. We see that the effect on N_c of a given net-cycle input state I_0 is to select an n-input switch at each position in N_c, the result being an n-delay input-independent cycle N_{I_0}. If $I(t) = I_0$ then the behavior of the cycle N_c for that one moment of time will be exactly that of N_{I_0}.

The importance of the preceding observation lies in the relation between the transition graphs of the various $N_I(t)$ and the transition graph of the cycle N_c. Let G be the transition graph of N_c and let the 2^k possible net-cycle input states of N_c, $(h_0,\ldots,h_{k-1})$, be labelled $I_0, I_1,\ldots,I_{2^k-1}$. Let G_{I_j} be the subgraph of G obtained by retaining all of the vertices of G and only the edges of G labelled I_j (see Part 2). Then G_{I_j} is exactly the transition graph of the net-cycle N_{I_j}, the input-independent net-cycle selected when $I(t) = I_j$. Conversely, if the 2^k graphs G_{I_j}, $j = 0,\ldots,2^k-1$ are given, then the graph G can be constructed. This is done by simply superposing all the graphs G_{I_j} so that vertices with the same label are identified and each edge for each G_{I_j} appears in the result, G, connecting the same vertices. In the process of forming G from the G_{I_j} many of the properties common to all the G_{I_j} will reappear as properties of G.

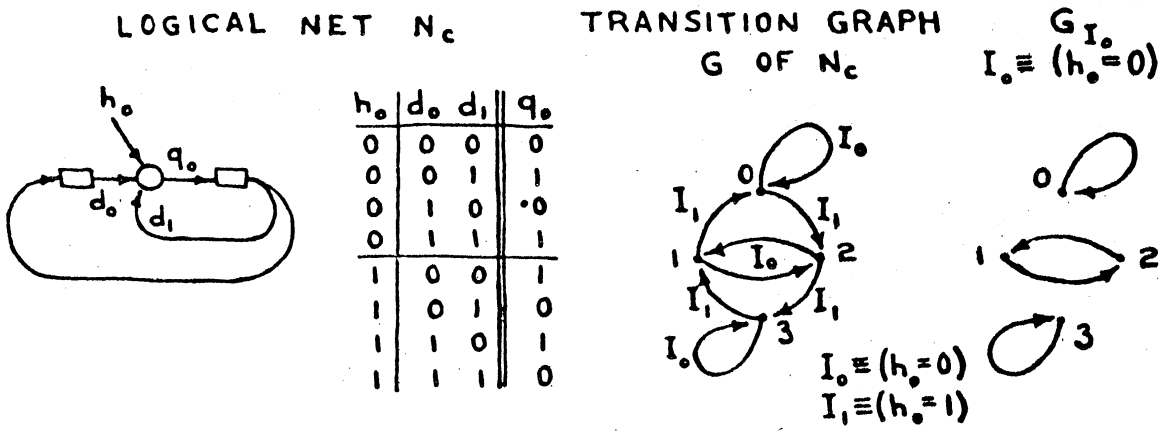


Fig. 13 An Example of the Selection of $G_{I(t)}$ By the Input-State I(t)

In the special case that a subgraph G_{I_j} of the transition graph G consists just of state-cycles, it can be conveniently represented by an element of the group of permutations on 2^n elements. To do this we make use of the fact that any permutation can be given by the product of a set of disjoint circular permutations: Let C_i be the i^{th} state-cycle of G_{I_j} (under some arbitrary ordering). Let $s_{i,0}, s_{i,1},\ldots,s_{i,n_i}$ be the net-states belonging to C_i ordered so that $s_{i,y+1}$ succeeds $s_{i,y}$ (and $s_{i,0}$ succeeds s_{i,n_i}) in G_{I_j}. The circular permutation $(s_{i,0}\ s_{i,1} \cdots s_{i,n_i})$ represents the state-cycle C_i and, thus, the group element

$$g_j = (s_{0,0}\ s_{0,1} \cdots s_{0,n_0})(s_{1,0} \cdots s_{1,n_1})\ldots(s_{v,0} \cdots s_{v,n_v})$$

represents G_{I_j}. Note that, if the state cycles C_i are normal state-cycles, the operation of inversion on the pair of states s_{2h} and s_{2h+1} can be represented by multiplying g_j on the left by the transposition $(s_{2h}\ s_{2h+1})$. Thus, if the switch in a locally-balanced cycle is obtained by inversions on pairs $h_0,h_1,\ldots,h_r$ of a normally oriented switch, the resulting transition graph G_{I_j} will be represented by the group element

$$g'_j = (s_{2h_0}\ s_{2h_0+1})(s_{2h_1}\ s_{2h_1+1})\ldots(s_{2h_r}\ s_{2h_r+1}) \cdot g_j$$

where g_j represents a transition graph consisting just of normal state-cycles.

If G is the transition graph of a net-cycle N_c with input and if each subgraph G_{I_j} of G consists just of state-cycles, then given an input state sequence of period m the resulting net-state period of N_c can be determined by means of the group representation. This is accomplished by using the graphs $G_{I(0)}, G_{I(1)},\ldots,G_{I(m-1)}$ specified by input states $I(0), I(1),\ldots,I(m-1)$, where $I(t) = I(j)$ if and only if $t \equiv j \bmod m$, $j = 0,\ldots,m-1$. If $g_{I(t)}$ is the group element corresponding

to $G_{I(t)}$, then the net-state period will be a divisor of the product $m \cdot r_g$, where r_g is the order of the group element

$$g = g_{I(0)} \cdot g_{I(1)} \cdots g_{I(m-1)}.$$

As a more general example of the way properties of the G_{I_j} reappear in G, I will show how Theorem 11 can be applied to net-cycles in general after the definition of a cycle of order r (given in Part 4) is extended appropriately. Let E_1 be an arbitrary switching element with b = n+k inputs. Let n of these inputs, $p_0,\ldots,p_{n-1}$ be identified with the outputs of elements belonging to a cycle N_c. Let k of the inputs $h_0,\ldots,h_{k-1}$ be identified with the outputs of elements not belonging to N_c. E_1 will be said to be of <u>order r w.r.t. a cycle</u> N_c if, ignoring the argument columns $h_0,\ldots,h_{k-1}$, it is of order r when just columns $p_0,\ldots,p_{n-1}$ are considered. A net-cycle N_c, with or without net-cycle inputs, will be an <u>(n,r)-cycle</u> just in case the normal form of the cycle contains exactly n delays, at least one switch of order r w.r.t. the cycle N_c, and no switches of proper order $r' > r$ w.r.t. the cycle. An (n.r)-cycle will, in effect, have no switch which receives more than r distinct feedbacks from delays in the cycle

<u>Theorem 12.</u> The set of state-cycle partitions of the set of all (n,r)-cycles properly includes the set of state-cycle partitions of any collection of (n',r')-cycles with $n' \leq n$, $r' < r$.

Proof:

1) In order to apply Theorem 11 to an arbitrary net-cycle, N_c, of order r, consider first the transition graph G_{I_j} of N_c. Each G_{I_j} will be the transition graph of an input-independent cycle N_{I_j} of order r (since the switches of N_{I_j} cannot have any cycle feedbacks not present in N_c). Thus each G_{I_j} will be subject to Theorem 11 as applied to input-independent

net-cycles of order r. The result will be that each G_{I_j} can satisfy the following three conditions:

1.1) G_{I_j} consists just of disjoint state-cycles.

1.2) All normal state-cycles with more than n-r+1 ones are present in G_{I_j}.

1.3) One normal state-cycle with n-r+1 ones is not present in G_{I_j}.

2) It follows from step (1) and the discussion preceding this theorem that there is a cycle N_c of order r with the following properties for its transition graph G:

2.1) All subgraphs G_{I_j} of G consist only of disjoint state-cycles.

2.2) For each state s with i $>$ n-r+1 digits equal to 1 there is a unique state s' with i digits equal to 1 which succeeds s no matter what the input state I(t) is. The cycle of states so determined has n states, or a divisor thereof, as elements.

2.3) There is a state s_0 with i_0 = n-r+1 digits equal to 1 which, for some input state I_0, has a successor state s'_0 with i_0-1 digits equal to 1. The state-cycle to which s_0 belongs has $n_0 > n$ elements.

3) Using the state-cycle partition (see the definition preceding Theorem 9 - noting that a state-cycle in the transition graph is defined analogously to a net-cycle in a logical net) properties (2.1)-(2.3) can be restated:

3.1) The state-cycle partition of G contains no subset of elements not belonging to a state-cycle.

3.2) Each net-state with more than n-r+1 digits equal to 1 belongs to a subset of the partition which contains exactly n states or a divisor thereof.

3.3) Some net-state with n-r+1 digits equal to 1 belongs to a subset with more than n states.

4) For an n-delay cycle N'_c of order $r' < r$ no derived transition graph G'_{I_j} can satisfy all three conditions (1.1)-(1.3) that each G_{I_j} satisfies. Hence no n-delay net-cycle of order $r' < r$ can have a transition graph satisfying the conditions (2.1)-(2.3). Thus, in turn, no n-delay net-cycle of order $r' < r$ can exhibit the state-cycle partition of step (3).

5) The remainder of the proof follows the argument of step (12) of Theorem 11.

Theorem 12 can be interpreted in Moore's framework in much the same way Theorem 11 was. Let $B_j(n,k)$ be a black box having the following properties:

1) n observable outputs each of which is a delay element output,

2) k inputs (net inputs) whose states at t = 0,1,2,... are specified by the observer,

3) an arbitrary number of elements in the box all belonging to one and the same (n,r)-cycle.

The set of behaviors possible for the set of all $B_j(n,k)$ which contain an (n_0,r_0)-cycle properly includes the set of behaviors possible for any collection of $B_j(n,k)$ which contain an (n_1,r_1)-cycle such that $n_1 \leq n_0$ and $r_1 < r_0$. In other words, no cycle with at most n delays or k' inputs and less than r feedbacks to each switch can imitate the behavior of particular cycles with n delays, k inputs and r feedbacks to one or more switches.

6. Comments and Conjectures

The central purpose of this paper has been to present a set of operations and methods particularly suited to the investigation of cycles in logical nets. The outcome has been an interlocking sequence of definitions and operations leading from the simplest cycles to cycles of arbitrary complexity:

Type of net-cycle	Definitions providing for extension to next level	Operations generating extension to next level
1) simple cycle without input	locally-balanced switch; derived transition table; normal state-cycle	inversion
2) locally-balanced cycle	order of switch	unbalancing
3) input-independent net-cycle with one switch	normal form of net-cycle; derived transition table for normal form	finite induction on number of switches in normal form of net cycle (listing possible effects of added switches on derived transition table)
4) general input-independent net-cycle	constant input subgraphs, G_{I_j}, of transition graph G	selection operation of input-state I(t) (selects $G_{I(t)}$ from G).
5) general net-cycle with input		

Because of the way in which a net-cycle is defined each element in a net can belong to at most one net-cycle. It follows from this that the

cycles in a logical net can be ranked and therefore that the cascading operation is sufficient to generate any logical net (cf. Burks-Wang [1] and Part 2 of the present paper). Hence an additional row can be added to the table although the present paper does not specifically consider the case (except for simple cycles in Part 3):

5) general net-cycle with input	rank of net-cycle	cascading of net-cycles and n.-c. nets
6) logical nets in general		

In this table the class of all net-cycles of a given type properly includes preceding types of net-cycle. Generally, the operations presented in the table and discussed throughout the paper are useful in computing the behavior of particular net-cycles as well as in proving theorems about the various types of net-cycle.

Using periodic input as a tool one can often prove theorems concerning a given class of nets which would be difficult to prove in any other way. For example, in Part 3 periodic input was used to show that cycles in logical nets provide more than just storage. Incidentally results there show that a particular case of a conjecture of Burks-Wang [1;p.292] is true. The conjecture is: For any degree d, there is some transformation not realized by any net of degree d - a net is of degree d if it contains at least one cycle of degree d and none of higher degree; a cycle is of degree d if it contains d delays. We see thus, by Part 3, that there are transformations on periodic input-state sequences not accomplished by any net of degree 1. The results of Parts 4 and 5 lend strong support to the following stronger conjecture:

For any (n,r) there is some transformation not realized by any net containing only (n,r)-cycles.

In fact it should be possible to construct a lattice of behavioral transformations defined as follows: Let $N_{(n,r)}$ be the set of all logical nets containing only (n,r) cycles. With each logical net in $N_{(n,r)}$ will be associated a transformation which gives the net state sequence produced by each input state sequence. Let $B_{(n,r)}$ be the set of transformations associated with the set $N_{(n,r)}$. The lattice should satisfy the following conditions:

1) $B_{(n,r)}$ properly includes $B_{(n',r')}$ if $n' < n$ and $r' \leq r$ or if $n' \leq n$ and $r' < r$;

2) g.l.b. $[B_{(n_1,r_1)},B_{(n_2,r_2)}] = B_{(n_0,r_0)}$ where $n_0 = \min(n_1,n_2)$ and $r_0 = \min(r_1,r_2)$;

3) l.u.b. $[B_{(n_1,r_1)},B_{(n_2,r_2)}] = B_{(n_3,r_3)}$ where $n_3 = \max(n_1,n_2)$ and $r_3 = \max(r_1,r_2)$.

It seems that the interrelations between periodic input and net state sequences, net cycles, state cycles, and permutation cycles, as sketched in Parts 3,4, and 5, will provide most of the operations and methods needed for the proof of this conjecture.

REFERENCES

1. Burks, Arthur W., and Hao Wang, "The Logic of Automata", J. Assn. Computing Machinery, 4: 193-218 and 279-297 (1957).
2. Burks, Arthur W. and Jesse B. Wright, "Theory of Logical Nets", Proc. IRE, 41: 1357-1365 (1953).
3. Copi, Irving M., Calvin C. Elgot, and Jesse B. Wright, "Realization of Events by Logical Nets", J. Assn. Computing Machinery, 5: 181-196 (1958).
4. Kleene, Stephen C., "Representation of Events in Nerve Nets and Finite Automata", pp. 3-41 in Automata Studies, edited by C. E. Shannon and J. McCarthy, Princeton Univ. Press, 1956.
5. Mealy, George, "A Method for Synthesizing Sequential Circuits", Bell System Tech. J., 34: 1045-1079 (1955).
6. Moore, Edward F., "Gedanken-Experiments on Sequential Machines", pp. 129-153 in Automata Studies, edited by C. E. Shannon and J. McCarthy, Princeton Univ. Press, 1956.

INDEX OF DEFINITIONS, SYMBOLS (FIRST APPEARANCE), AND THEOREMS