CYCLES IN LOGICAL NETS*

BY

JOHN H. HOLLAND [1]

ABSTRACT

This paper investigates the influence of cycles in a logical net upon the complexity of its behavior. The investigation is mainly concerned with two questions:

1. A logical net with a periodic input sequence produces a periodic output sequence; how is the spectrum of periodic outputs related to the level of cycle complexity?

2. Is there a level of complexity c (suitably defined) such that any behavior possible for a fixed logical net can be realized by a logical net constructed only of cycles of complexity $c' \leqslant c$? The first and more difficult question is fully answered only in the case of nets constructed of cycles having a feedback coefficient $r = 1$ (suitably defined). The second question is answered in the negative for individual cycles and it is conjectured that a similar answer holds for nets in general.

1. INTRODUCTION

Cycles in a logical net in general have a profound effect upon the net's behavior. This paper investigates the relation between the complexity of such cycles and the complexity of the behavior that results. Much of the investigation is concerned with two questions:

1. A logical net with a periodic input sequence produces a periodic output sequence; how is the spectrum of periodic outputs related to the level of cycle complexity?
2. Is there a level of complexity c (suitably defined) such that any behavior possible for a fixed logical net can be realized by a logical net constructed only of cycles of complexity $c' \leq c$?

The first and more difficult question is fully answered only in the case of nets constructed of cycles having a feedback coefficient $r = 1$ (suitably defined); Theorem 3 provides the answer for this case. The final theorem of the paper leaves the second question still unanswered for nets in general, but provides the following answer for individual cycles: If we consider the set C of all cycles having $n' \leq n$ delays and a feedback coefficient $r' < r$, then there is a cycle having n delays, a feedback coefficient r, and a state transition diagram which cannot be

* The author would like to thank Arthur Burks and Jesse Wright for discussions on many topics related to this paper.

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

[1] Logic of Computers Group, The University of Michigan, Ann Arbor, Mich.

realized by any cycle in C. Equating cycle complexity to the number pair (n', r'), and defining an ordering (a lattice) such that $(n', r') < (n, r)$ just in case $(n' < n,\ r' \leq r)$ or $(n' \leq n,\ r' < r)$, we can answer the second question in the negative for individual cycles.

The theorem last mentioned rests upon an interlocked sequence of definitions, operations, and theorems leading from the simplest input-free cycles to arbitrary cycles with input. The operations in particular play a key part in the development; they provide a means of obtaining information about state cycles in the state transition diagram of an arbitrary net cycle in terms of the transition diagrams of simple input-free cycles. Aside from this role, these operations have application in a wide range of problems and questions concerning the relation of cycle complexity to complexity of behavior. Two simple examples of such applications are given at convenient places in the development; one application is to a conjecture of Burks and Wang (**1**)[2] and the other is to a problem investigated by de Bruijn (**3**).

The development proceeds in the following stages:

Section 2 defines the term "cycle" as used in this paper and shows the existence of a normal form for cycles. Intuitively, a cycle with n delays has a normal form in which there are also exactly n switches, with the output of the jth delay going to the jth switch (and perhaps others if the cycle is complex) which in turn acts as input to the $(j + 1)$th delay (modulo n). This normal form later makes possible a relatively simple definition of the order of a net cycle (the feedback coefficient mentioned earlier) and considerably simplifies the proofs by drastically reducing the number of cases and variations to be considered.

Section 3 considers the effect of periodic input upon the behavior of nets in which all cycles are of order 1 (each delay in the normal form of each cycle acts as input to exactly one switch in that cycle). This part of the investigation was motivated by the following consideration: If it could be shown that cycles only serve to provide logical nets with memories of various recycling times, then it would be possible to accomplish this function by means of cycles of order 1 ("simple cycles"). Under such conditions there would be little need to consider cycle complexity further except insofar as to show how to reduce the complex cycle to an equivalent simple cycle. In fact this conjecture proves to be untrue as is shown in a Corollary to Theorem 3. Theorem 3, besides giving a complete description of the output spectra for nets composed of simple cycles, shows that no such net can realize the behavior of a 2-delay cycle of order 2. Section 3 concludes by showing that a particular case of a conjecture of Burks and Wang (**1**) is true.

The sequence of theorems in Section 4 prepares the way for the results given in the last section. Section 4 begins by considering input-

[2] The boldface numbers in parentheses refer to the references appended to this paper.

free simple cycles with one switch and proceeds step-by-step to the general input-free cycle. At each level an operation is developed which yields information about the state transition diagrams of cycles at the next level in terms of properties established for the given level. That is, each theorem (with the exception of Theorem 8) provides information which when combined with the appropriate operation yields information appropriate to the next step. In Section 5 the results for input-free cycles are extended to the case of the general cycle with input, yielding the result (Theorem 12) mentioned earlier. This interlocking sequence of definitions and operations leading from input-free simple cycles to the general cycle with input can be summarized as shown in Table I (the reader is referred to context for exact definitions).

TABLE I.

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-free 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-free net cycle	constant input subgraphs, G_{I_i}, of transition graph G	selection operation of input-state $I(t)$ (selects $G_{I(t)}$ from G).
(5) general net cycle with input	rank of net-cycle	cascading of net cycles and net-cycle nets
(6) logical nets in general		

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 Section 2 of the present paper). Hence, row (6) was added to the table although the present paper does not specifically consider the case (except for simple cycles in Section 3).

In Table I, 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.

In Section 4, Theorem 8 provides a constructive solution in terms of input-free locally balanced cycles of a problem investigated by de Bruijn (3). The proof illustrates the use of the operations just summarized in exploring questions other than the two posed at the outset. In Section 5, preceding Theorem 12, a method is given for computing the periodic input/periodic output relation of any given backwards deterministic cycle. The computation makes use of the properties of a related set of input-free locally balanced cycles. The paper concludes with a conjecture that the result of the final theorem holds for nets in general; that is, it is conjectured that the answer to the second question asked earlier is negative.

2. UNDERLYING CONCEPTS

The logical nets considered in this paper will be well-formed logical nets, essentially as defined by Burks and Wang in Part 2.3 of their paper (1). The nets are based upon a set of primitives consisting of a delay element and an infinite class of switching elements. Terminology, except where explicit definitions are given, will be that of Burks and Wang. The state transition graph will be used as a means of compactly picturing the behavior of a logical net. E. F. Moore's conventions (6) will be used except that the output associated with each net state will not in general be indicated.

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, $E_1 \underline{\mathrm{d}}\, E_2$, if and only if the output of E_1 is connected to one of the inputs of E_2. A sequence of elements $F_1, \cdots, F_n$ is a *drive sequence from* F_1 *to* F_n if and only if $F_j \underline{\mathrm{d}}\, F_{j+1}$ for $j = 1, \cdots, n-1$. An element E_1 *drives* an element E_2, $E_1 \underline{\mathrm{D}}\, E_2$, 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 $E_1 \underline{\mathrm{D}}\, E_1$. This cycle consists of the set of all elements, E_j, such that both $E_1 \underline{\mathrm{D}}\, E_j$ and $E_j \underline{\mathrm{D}}\, E_1$. Or, more formally, a set of elements, C, is a *cycle* if and only if, for all E_i, E_j in C, (1) $E_i \underline{\mathrm{D}}\, E_j$ and $E_j \underline{\mathrm{D}}\, E_i$ 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) $E_i \underline{\mathrm{D}}\, E_j$ and $E_j \underline{\mathrm{D}}\, E_i$ and (2) for each relation $E_i \underline{\mathrm{D}}\, E_j$ 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.

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 $r - 1$.

Any cycle can be reduced to a much simpler *normal form*: if the cycle has n delays then there is an equivalent normal form which has at most n switches, so arranged that the output of delay j goes to switch

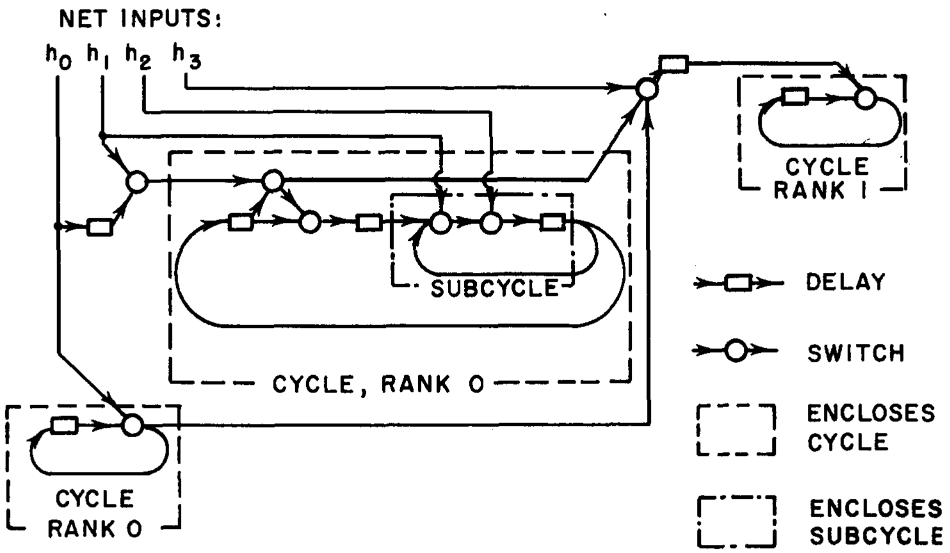


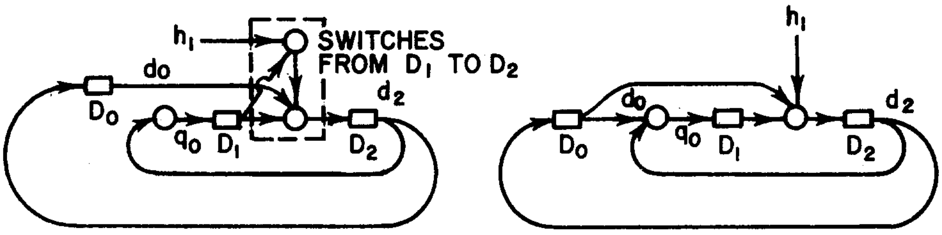
FIG. 1. A logical net with cycles marked.

j (and perhaps others) and the output of switch j acts as input to delay $j + 1$ (modulo n). To establish this normal form the following definitions are required: 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.

Let the set of delays belonging to a given cycle be ordered and labelled $D_0, D_1, \cdots, D_{n-1}$. Since we are dealing with well-formed nets and since each delay has but one input, there must be at most one switch S_j which directly drives delay D_j. If $D_{j-1(\text{mod } n)} \not\underline{s} D_j$ simply add a new input to S_j with the proviso that the output of the switch is to be independent of this new input (that is, the output of the switch is still uniquely determined by the inputs initially given). By connecting the output of $D_{j-1(\text{mod } n)}$ to this new input we have formally the result $D_{j-1(\text{mod } n)} \underline{s} D_j$. Let this be done for all j, $j = 0, 1, \cdots, n - 1$.

For each j, consider *all* D_i in the cycle such that $D_i \underline{s} D_j$ (one such D_i will now be $D_{j-1(\text{mod } n)}$). Among all the switches belonging to one or more of the sets of switches from the $\{D_i\}$ to D_j there will be a total of k switch inputs not identified with the output of any other switch in these sets or with the outputs of the $\{D_i\}$. It can easily be seen that each assignment of states to these k switch inputs uniquely determines the input state of D_j when the output state of each D_i driving D_j via switches is given. Thus, assuming that there are m such D_i, we can replace all of the aforementioned switches by a single $(k + m)$ input switch. Of the inputs to this new switch, k will be identified as inputs to the cycle corresponding to the k selected inputs of the given switches. The other m inputs will be connected to the outputs of the m delays D_i. The output of the new switch will be connected to the input of D_j. Once this is done for each D_j, the normal form of the cycle results.

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.



d_2	q_0
0	δ_0
1	δ_1

$$q_0(t) \equiv f_0(d_2(t))$$

d_0	d_2	q_0
0	0	δ_0
0	1	δ_1
1	0	δ_0
1	1	δ_1

$$q_0(t) \equiv f_0(d_2(t))$$

FIG. 2. A cycle and its normal form.

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, that is, cycles with no proper subcycles. A cycle of this type, which I will call a *simple cycle*, can be more 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 (see Fig. 3).

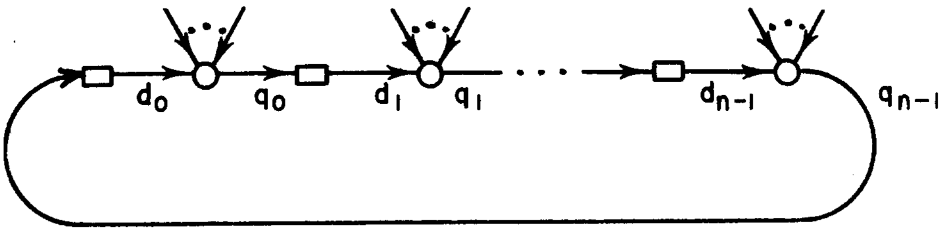


FIG. 3. Normal form of simple cycle.

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 has the same behavior. The conjecture is plausible on the view that the simple cycles provide "delay line" or "reverberatory" storage of various periods while the rest of the net provides encoding, switching, decoding, and other logical operations. The following three theorems show that this conjecture is in fact false. The last theorem of the section, Theorem 3, shows a good deal more than this— for any net composed of simple cycles, it characterizes the spectrum of net state periods resulting from periodic input sequences.

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 outline[3]

1. Consider a particular delay, i. Let $d_i(t)$ be the state of this delay at time t. If $d_i(t_1) = d_i(t_2)$ and if the input sequence $I(t_1 + 1)$, $I(t_1 + 2), \cdots, I(t_1 + k)$ is identical to the input sequence $I(t_2 + 1)$, $\cdots$, $I(t_2 + k)$ for some $k = jn$, then $d_i(t_1 + k) = d_i(t_2 + k)$. (Consideration shows that this statement is not true in general for $k \neq jn$.) Under the same conditions if $d_i(t_1) = 1 - d_i(t_2)$ then $d_i(t_1 + k) = 1 - d_i(t_2 + k)$.

2. The input sequence repeats every m steps by hypothesis. Thus, for $k = jn$, the input sequence $I(t_1 + 1), \cdots, I(t_1 + k)$ is identical to the sequence $I(t_1 + k + 1), \cdots, I(t_1 + 2k)$ for j such that $k = jn = \text{l.c.m.}\ (m, n)$.

3. Since $d_i(t)$ can take on only two values, we must have by (1) and (2) either

$$d_i(t_1) = d_i(t_1 + \text{l.c.m.}\ (m, n))$$

or

$$d_i(t_1) = 1 - d_i(t + \text{l.c.m.}\ (m, n)) = d_i(t + 2\ \text{l.c.m.}\ (m, n)).$$

Thus, for $p = 2$ l.c.m. (m, n), we must have

$$d_i(t_1) = d_i(t_1 + jp) \qquad \text{for} \qquad j = 0, 1, 2, \cdots.$$

4. Step (3) holds for each delay, hence the net state sequence must repeat with period p.

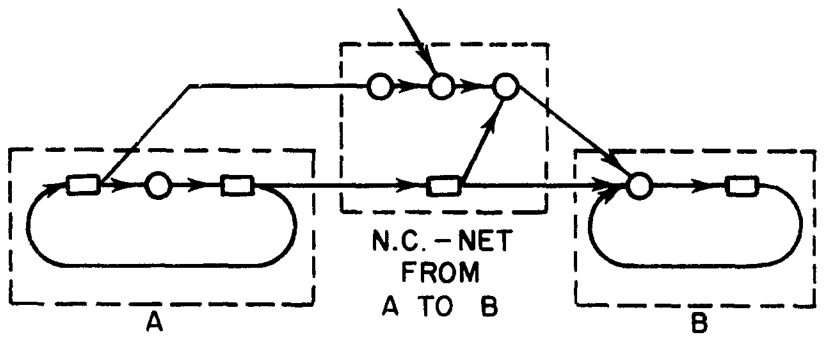


FIG. 4. An n.c.-net.

In nets constructed with more than one simple cycle, the simple cycles can be separated by subnets having no cycles. The following definitions are intended to give a precise interpretation to this statement: A drive sequence from an element F_1 to an element F_2 will be

[3] At the suggestion of the JOURNAL reviewers, the proofs of this theorem and of Theorems 5, 7, 8, 9, 10 and 11 have been omitted. They may be found in the microfilm edition of the author's thesis, "Cycles in Logical Nets," Ph.D. dissertation, 1959, The University of Michigan.

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.*

Theorem 2. If the sequence of input states to a net without cycles (for example, an n.c.-net) has a proper period m, then the net state sequence of the 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 be 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; for example, 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: 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 $n_1, \cdots, n_k$ 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. $(m, n_1, \cdots, n_k)$, 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 (that is, 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 Section 2). Any element driving C but not in 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 driving cycle inputs 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 $j = -1, 0, 1, \cdots, r_0$, 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; and (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 $n_0, \cdots, n_{k_0-1}$ 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), $i = 0, \cdots, k_0 - 1$. 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

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

5. Let $n_0, \cdots, n_{k_0-1}, n_{k_0}, \cdots, n_{k_1-1}, \cdots, n_{k_j-1}$ 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. $(m, n_0, \cdots, n_{k_{j-1}-1})$. 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

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

Applying the reasoning of step (4) we see then that the proper net state period of N_j will be a divisor of l.c.m. $(m, 2^{j+1}$ l.c.m. $(m, n_0, \cdots, n_{k_{j-1}})$.

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

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

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

The corollary shows that no net composed of simple cycles can, for example, act as a ternary (base 3) counter. To do so it would have to be in some distinguished state S_0 for every third occurrence of the distinguished input state I_0. This condition holds *a fortiori* for a periodic input sequence, whence we would require $p \equiv 0 \bmod 3$, contradicting the corollary. Since there exists a 2-delay logical net which can count base 3, the corollary at once establishes the falsity of the conjecture stated at the beginning of this section.

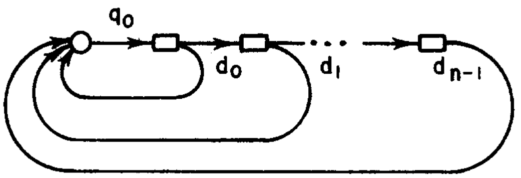
If this result is not surprising, it at least shows the oversimplification present in the idea that the main function of cycles in a system is to provide "memory" or storage of information. Here we have systems with any number of cycles of arbitrary lengths (arbitrary recycling times) which have a very 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.

The results here also show that a particular case of a conjecture of Burks and 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. Since nets of degree 1 must be composed of simple cycles, we see that there are transformations on periodic input-state sequences not accomplished by any net of degree 1. By 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.

4. INPUT-FREE CYCLES

The results of Section 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.

The present section will start out by relating properties of the state-transition graph to changes in feedback in a class of input-free 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. The two kinds of cycles will be termed 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.



d_0	d_1	$\cdots$	d_{n-2}	d_{n-1}	q_0
0	0	$\cdots$	0	0	ϵ_0
0	0	$\cdots$	0	1	ϵ_1
0	0	$\cdots$	1	0	ϵ_2
0	0	$\cdots$	1	1	ϵ_3
$\cdots$		$\cdots$	$\cdots$		$\cdots$
1	1	$\cdots$	1	0	ϵ_{2^n-2}
1	1	$\cdots$	1	1	ϵ_{2^n-1}

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

FIG. 5. A locally balanced cycle and the truth table of its switch.

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 $j = 0, 1, \cdots, 2^{n-1} - 2$, 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, $j = 0, 1, \cdots, 2^{n-1} - 2$. 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}, $j = 0, \cdots, n-2$.

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

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 $(p_0(t), \cdots, p_{n-1}(t))$ of the states of the n inputs to the switch in a locally balanced cycle. By the definition of a locally balanced cycle, this n-tuple is identified with the ordered n-tuple of delay output states at time t, $(d_0(t), \cdots, d_{n-1}(t))$. 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 jth 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,

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

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 the *derived transition table* for the locally balanced cycle (Table II).

TABLE II.—*The Derived Transition Table of a Locally Balanced Cycle.*

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

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

3. Now, the j^{th} pair of the locally balanced switch gives rise to a pair of successor n-tuples, $s_{2j}(t+1) = (\epsilon_{2j}, p_0(t), \cdots, p_{n-2}(t))$ and $s_{2j+1}(t+1) = (\epsilon_{2j+1}, p_0(t), \cdots, p_{n-2}(t))$. 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 (Table II). 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, permutations 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-free cycle consists only of disjoint state cycles just in case the cycle is backwards deterministic in the sense of Burks and Wang (1, p. 286). Using this fact, and noting that $\epsilon_{2j} = \epsilon_{2j+1}$ implies $s_{2j} = s_{2j+1}$, 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 $\epsilon_{2j} = 0$, $\epsilon_{2j+1} = 1$. A pair will be said to be *inversely oriented* if $\epsilon_{2j} = 1$, $\epsilon_{2j+1} = 0$. 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-free 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 $n_p = \dfrac{2^p - 2^{p'}}{p}$ 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.

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 $(0, 0, \cdots, 0)$ becomes 0, $(0, \cdots, 0, 1, 0)$ becomes 2, and $(1, 1, \cdots, 1)$ 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, that is, 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 with respect to E_1, then the transition graph with respect to 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 with respect to E_1, then the transition graph with respect to 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.

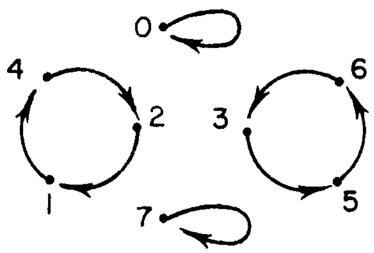
Case 1. s_{2j}, s_{2j+1} belong to different state cycles C_1, C_2.

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 — Derived Transition Table — Transition Graph

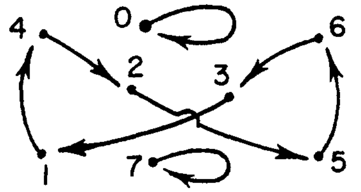
None

	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



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 $\epsilon_5 = 0$.

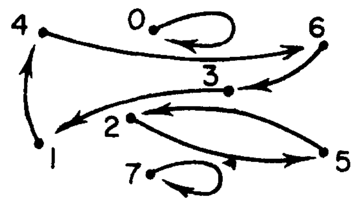


FIG. 6. Effects of inversions on the transition graph of a locally balanced cycle.

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.

The next theorem shows the application of the inversion operation to the solution of a combinatorial problem earlier investigated and solved by N. G. de Bruijn (3). The problem is to show, for any n, that there is an ordered cycle of 2^n digits 0 or 1 such that the 2^n possible ordered sets of n consecutive digits of that cycle are all different. Logical nets which can generate such sequences are of interest for error-correcting codes and for pseudo-random number generation. Theorem 8 not only proves the above statement, but in its proof (not included here) also shows how to construct input-free locally balanced cycles which, for any n, will generate such sequences as output.

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; that is, the net will have a net-state sequence of period 2^n.

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-free locally balanced cycles in which some of the feedback loops to the switch have been omitted, that is, cycles in which the switch receives $k \leqslant n$ inputs from the cycle.

Just before Theorem 5 it was noted 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, $0 \leqslant i_0 < \cdots < i_{k-1} \leqslant n - 1$, such that all arguments with the same values for $p_{i_0}, \cdots, p_{i_{k-1}}$ 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 $p_{i_0}, \cdots, p_{i_{k-1}}$; thus in a cycle the switch could be replaced by a switch with $k \leqslant n$ inputs identified with delay outputs $d_{i_0}, \cdots, d_{i_{k-1}}$.

For a locally balanced switch, $i_{k-1} = n - 1$ in the above definition. This is the case because, in each pair determined by giving values to $p_0, \cdots, p_{n-2}$, $q(t) = \epsilon_{2j}$ 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, $0 \leqslant i_0 < \cdots < i_{k-2} < n - 1$, such that all pairs with the same values for $p_{i_0}, \cdots, p_{i_{k-2}}$ have the same orientation.

TRUTH TABLE OF A SWITCH OF PROPER ORDER 2

P_0	P_1	P_2	q_0
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

⬚ ENCLOSES COLUMNS USED IN DEFINITION

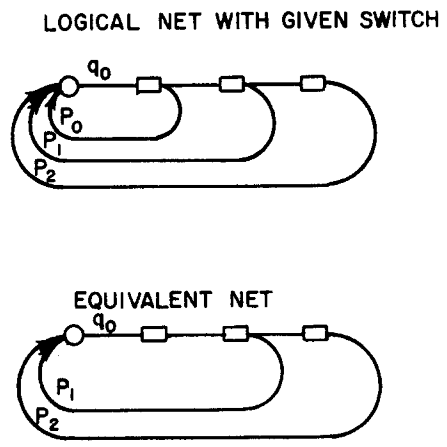


FIG. 7. A switch properly of order 2.

Theorem 9 will describe some of the changes occurring in the transition graph of a locally balanced cycle as the order of its switch is increased (that is, as the amount 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 $k' < k$.

Let a pair be called *unbalanced* 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-free cycle with one switch.

Theorem 10. The set of state cycles for a given input-free 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, or order $k' \geqslant k$. 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.

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

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-free cycles.

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

Intuitively, Theorem 11 says that for input-free cycles, there is no upper limit of complexity c such that the behavior of any given input-free cycle can be realized by some cycle of complexity $c' < c$. Here complexity is defined as a number pair (n, k), where n is the number of delays in the cycle, and k is the maximum of the numbers k_j, $j = 0, 1, \cdots, n - 1$, where k_j is the number of delays in the cycle which feed back their outputs to delay j.

We can look at this result in another way:

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-free 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. That is, for each level of complexity $c = (n, k)$ there are input-free cycles of complexity c which can be distinguished by the experiment from any of complexity $c' < c$. Moreover, the set of behaviors possible for black boxes of complexity (n, k) properly includes the set of behaviors possible for boxes having $n' \leq n$ delays and $k' < k$ feedbacks to each switch.

Theorem 12 in the next section makes direct use of Theorem 11 to prove the same statements for cycles with input.

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-free 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, that is, 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 $h_0, \cdots, h_{k-1}$ corresponding to the k net-cycle inputs and n argument columns $p_0, \cdots, p_{n-1}$ corresponding to the n net-cycle delay outputs. If a given switch in N_c has $j < n$ inputs identified with delay outputs $d_{i_0}, \cdots, d_{i_{j-1}}$ of N_c then the truth table output $q(t)$ will of course depend only upon the j columns $p_{i_0}, \cdots, p_{i_{j-1}}$ of the n columns $p_0, \cdots, p_{n-1}$. That is, with respect to the n columns $p_0, \cdots, p_{i_{j-1}}$, 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 $h_{v_0}, \cdots, h_{v_{b-1}}$ of the k columns $h_0, \cdots, h_{k-1}$. 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 $h_0, \cdots, h_{k-1}, p_0, \cdots, p_{n-1}$. 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 $q_{n-1}, q_0, q_1, \cdots, q_{n-2}$ at the right of the $k + n$ argument columns the result

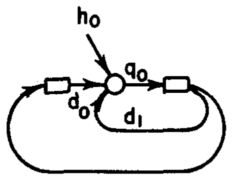
TABLE III.—*Derived Transition Table for a General Cycle with Input.*

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

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

Now fix a value, I_0, for $I(t) = (h_0(t), \cdots, h_{k-1}(t))$ and consider, in the derived transition table for N_c, the 2^n rows having the given values for the arguments $h_0(t), \cdots, h_{k-1}(t)$. 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-free cycle in which the switch at position q_i has, for arguments $p_0(t), \cdots, p_{n-1}(t)$, an output $q_i(t) = \epsilon_{i,x\cdot 2^n+y}$, where x = the decimal equivalent of the binary number $h_0(t)\cdot 2^{k-1} + \cdots + h_{k-1}(t)\cdot 2^0$ and y = the decimal equivalent of $p_0(t)\cdot 2^{n-1} + \cdots + p_{n-1}(t)\cdot 2^0$. That is, at each position in N_{I_0} we have a switch whose output for an argument $(p_0(t), \cdots, p_{n-1}(t))$ is simply the value $q_i(t)$ given for the corresponding n argument values of $p_0, \cdots, p_n$ 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-free 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}.

LOGICAL NET Nc



ho	do	d1	qo
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

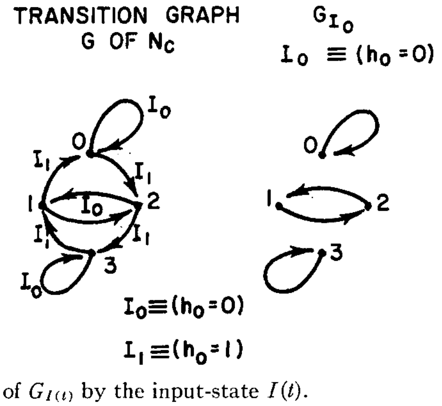


FIG. 8. An example of the selection of $G_{I(t)}$ by the input-state $I(t)$.

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, $(h_0, \cdots, h_{k-1})$, be labelled $I_0, I_1, \cdots, I_{2^k-1}$. 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 Section 2). Then G_{I_j} is exactly the transition graph of the net cycle N_{I_j}, the input-free net cycle selected when $I(t) = I_j$. Conversely, if the 2^k graphs G_{I_j}, $j = 0, \cdots, 2^k - 1$ 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.

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 $s_{i,0}, s_{i,1}, \cdots, s_{i,n_i}$ 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

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

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 $h_0, h_1, \cdots, h_r$ of a normally oriented switch, the resulting transition graph G_{I_j} will be represented by the group element

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

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 $G_{I(0)}, G_{I(1)}, \cdots, G_{I(m-1)}$ specified by input states $I(0), I(1), \cdots, I(m-1)$, where $I(t) = I(j)$ if and only if $t \equiv j \bmod m$, $j = 0, \cdots, m-1$. 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, Theorem 11 will be applied to net cycles in general. First, the definition of a cycle of order r (given in Section 4) must be extended appropriately: Let E_1 be an arbitrary switching element with $b=n+k$ inputs. Let n of these inputs, $p_0, \cdots, p_{n-1}$ be identified with the outputs of elements belonging to a cycle N_c. Let k of the inputs $h_0, \cdots, h_{k-1}$ be identified with the outputs of elements not belonging to N_c. E_1 will be said to be of *order r w.r.t. a cycle* N_c if, ignoring the argument columns

$h_0, \cdots, h_{k-1}$, it is of order r when just columns $p_0, \cdots, p_{n-1}$ are considered. A net cycle N_c, with or without net-cycle inputs, will be an (n, r)-*cycle* 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.

Theorem 12. 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 $n' \leqslant n$, $r' < r$.

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-free 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-free 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} (all states of a given normal cycle have the same number of ones in their coded form; when this number is i the normal state cycle will be said to contain i ones—see proof of Theorem 9 for exact discussion); and (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'_{Ij} can satisfy all three conditions (1.1)–(1.3) that each G_{Ij} satisfies (by Theorem 11). 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 is the counterpart of Theorem 11 for arbitrary cycles with input. Intuitively it says that there is no level of complexity c such that any behavior possible for a cycle can be realized by a cycle of complexity $c' < c$ (cf. the second question asked at the beginning of Section 1).

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 $t = 0, 1, 2, \cdots$ 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 $n_1 \leqslant n_0$ 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.

The results of Sections 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 behaviorial 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 $r' \leqslant r$ or if $n' \leqslant n$ 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 this paper, could provide a basis for the proof of this conjecture.

REFERENCES

(1) ARTHUR W. BURKS AND HAO WANG, "The Logic of Automata," *J. Assn. Computing Machinery*, Vol. 4, pp. 193–218 and 279–297 (1957).
(2) ARTHUR W. BURKS AND JESSE B. WRIGHT, "Theory of Logical Nets," *Proc. IRE*, Vol. 41, pp. 1357–1365 (1953).
(3) N. G. DE BRUIJN, "A Combinatorial Problem," *Indagationes Math*, Vol. 8, pp. 461–467 (1946).
(4) IRVING M. COPI, CALVIN C. ELGOT AND JESSE B. WRIGHT, "Realization of Events by Logical Nets," *J. Assn. Computing Machinery*, Vol. 5, pp. 181–196 (1958).
(5) STEPHEN C. KLEENE, "Representation of Events in Nerve Nets and Finite Automata," pp. 3–41 in "Automata Studies," edited by C. E. SHANNON AND J. MCCARTHY, Princeton, Princeton University Press, 1956.
(6) EDWARD F. MOORE, "Gedanken-Experiments on Sequential Machines," pp. 129–153 in "Automata Studies," edited by C. E. SHANNON AND J. MCCARTHY, Princeton, Princeton University Press, 1956.