INFORMATION AND CONTROL 6, 399–415 (1963)

On Dynamic Switching in One-Dimensional Iterative Logic Networks*

WILLIAM L. KILMER

Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, Massachusetts

A sequential iterative network (SITN) is a cascade of identical finite automata such that the ith automaton receives an x_i input from the outside world and a y_i input from its left neighbor, and produces a z_i output to the outside world and a y_{i+1} output to its right neighbor. We prove three main theorems: (*1*) For every integer k there is a cell definition such that a corresponding SITN either can or cannot switch from equilibrium to a cycling condition (i.e., oscillation) following a single x_i change according as $n \leqq k$ or $n > k$, respectively; (*2*) there do not exist algorithms to tell whether a given cell definition admits of a SITN that can start from equilibrium and following a single x_i change either (a) switch into a cycling condition, or (b) put out a $z_i = 1$ during a switching transient; and (*3*) there do not exist algorithms to tell whether a given SITN cell definition must have every switching transient following a single x_i change from equilibrium either (a) die out a bounded number of cells to the right of the change, or (b) extend all the way to the SITN boundary. All theorems are proved constructively on finite-state diagrams, and (*2*) and (*3*) hinge on an embedding of Minsky's Post Tag system results into such diagrams. We conclude with several iterative network equivalence demonstrations.

I. INTRODUCTION

We consider a concatenation of n identical logic cells as shown in Fig. 1. The ith cell has associated with it the following: (*1*) a memory state variable s_i, with domain of values $a_1, a_2, \cdots a_m$; (*2*) an external (i.e., outside world) input variable x_i, with domain $b_1, b_2, \cdots b_p$;

* This research was supported by Air Force Cambridge Research Laboratories Contract No. AF19(604)-6619, under the auspices of The Montana State College Electronics Research Laboratory, Bozeman, Montana. The author is currently at the Research Laboratory of Electronics, M.I.T., on leave from Montana State College.

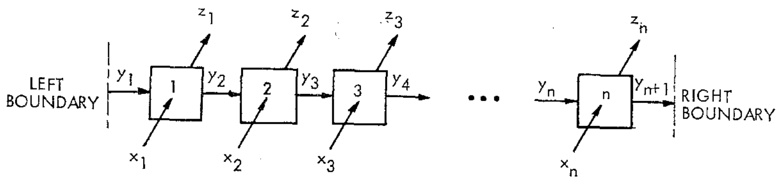


FIG. 1. Schematic diagram of a SITN

(*3*) lateral input and output variables y_i and y_{i+1}, respectively, each with domain $c_1, c_2, \cdots c_q$; and (*4*) an external output variable z_i, with domain $d_1, d_2, \cdots d_r$. We assume that the functions $z_i(x_i, y_i, s_i)$ and $y_{i+1}(x_i, y_i, s_i)$ are realized with zero time delay across the cells; and that the function $s_i(x_i, y_i, s_i)$ is realized with unit time delay within the cell. At time $t = 0$ the y_1, x_i, s_i variables are all assigned arbitrary values from their respective domains, and for all $t > 0$ the values of y_1 and all the x_i remain fixed. We denote such cell systems SITNs (for Sequential ITerative Networks).

An SITN is said to be in *equilibrium* at $t > 0$ if and only if all of its y_i, s_i values remain fixed from t on. A SITN is said to be *cycling* at $t > 0$ if and only if its over-all configuration of y_i, s_i values at t first recurs at $t + T$, for $T > 1$. If a SITN is in equilibrium at $t = -1$, and at $t = 0$ some y_1 and/or x_i values change, the corresponding sequence of y_i, s_i value changes is called a *transient* just in case the SITN reaches equilibrium at some $t > 0$. Otherwise the SITN enters a cycle.

The main purpose of this paper is to present some results on the problem of determining, for an arbitrary SITN cell definition, the various ways in which corresponding SITNs can switch from equilibrium to equilibrium, and equilibrium to cycle, following single x_i value changes. We also apply these results to non-SITN models discussed in Kilmer (1961, 1962a, b) in order to extend the present theory of switching dynamics for iterative logic networks.

II. ON THE RELATIONSHIP OF SITN SIZE TO POSSIBLE CYCLE ENTRY

In this section we give a constructive proof of Theorem 1.

THEOREM 1. *For every positive integer k there exists an SITN cell definition such that any corresponding n-celled SITN which is in equilibrium at $t = -1$, and which has a single x_i value change at $t = 0$, either can or cannot possibly enter a cycle at some $t \geq 0$ according as $n > k$ or $n \leq k$, respectively.*

Before embarking on the details of our proof, we first discuss its main idea. The plan is to design an SITN cell with two equilibrium "ground level" states, and $k - 1$ "ladder" states placed in a line between the ground level states and a pair of cycling states. Then we arrange our SITN cell so that if any x_i value change occurs in a corresponding SITN in equilibrium, the $(i + j)$th cell, for all $j < k$, first leaves its ground equilibrium state and successively mounts j ladder states, and then falls back to the opposite equilibrium state. Each such cell upon mounting its j ladder states carries its right neighbor

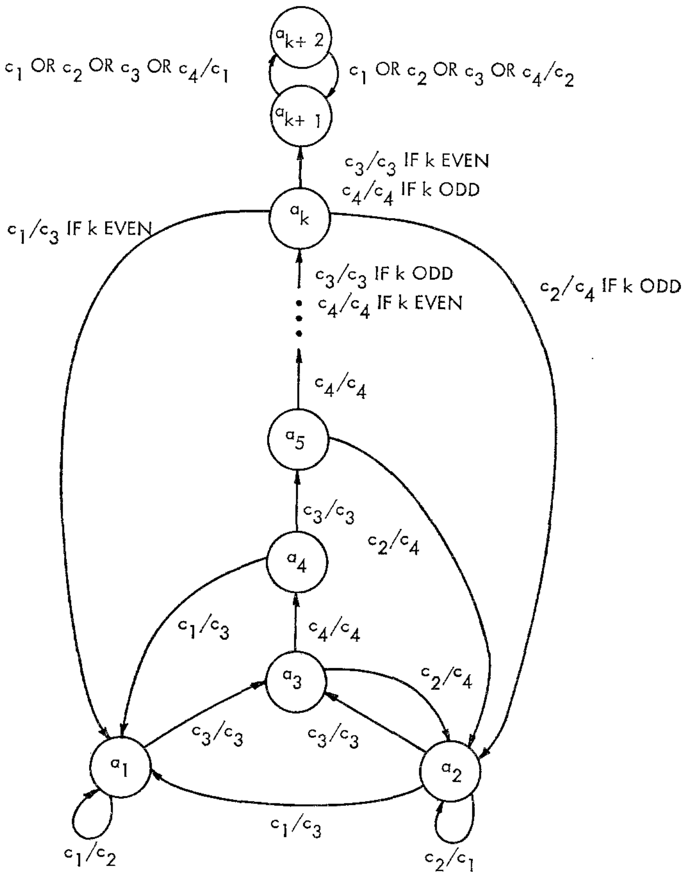


FIG. 2. Partially complete SITN memory state diagram for $x_i = b_1$ used in the proof of Theorem 1.

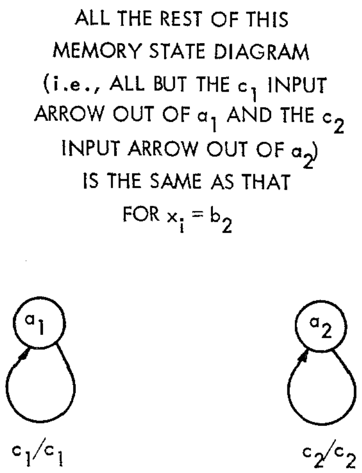

ALL THE REST OF THIS MEMORY STATE DIAGRAM (i.e., ALL BUT THE c_1 INPUT ARROW OUT OF a_1 AND THE c_2 INPUT ARROW OUT OF a_2) IS THE SAME AS THAT FOR $x_i = b_2$
a_1
a_2
c_1/c_1
c_2/c_2

FIG. 3. SITN memory state diagram for $x_i = b_2$ used in the proof of Theorem 1

cell up with it, and then kicks the neighbor cell up one more ladder state while it returns to its own new equilibrium state. Thus, by allowing the only possible cycle entry to occur at the kth ladder state, we insure that at least $k + 1$ cells are necessary in any corresponding SITN before it can enter a cycle following a single x_i value change from equilibrium.

We now proceed to the details of our proof. Consider the partially complete[1] SITN memory state diagram for $x_i = b_1$, shown in Fig. 2. The c_i/c_j label on each arrow there indicates that if a cell with x input equal to b_1 has the memory state value given at the tail of an arrow, and if its y input value is c_i, its corresponding y output value is c_j and its next memory state value is the one given at the head of the arrow.

Now assume an n-celled SITN in equilibrium at $t = -1$ as follows: all $x_i = b_1$; $s_i = a_1$ for i odd and $s_i = a_2$ for i even; and $y_1 = c_1$. That this is indeed an equilibrium is obvious from the self-returning arrows out of a_1 and a_2. Now suppose that at $t = 0$; x_1 changes to b_2. Figure 3 shows that this causes y_2 to change immediately from c_2 to c_1. Figure 2 accordingly indicates the successive SITN variable value changes listed in Fig. 4. Each square's entry in Fig. 4 contains the

[1] In the obvious sense that out of each s_i memory state value there should be an output arrow for each possible y_i value.

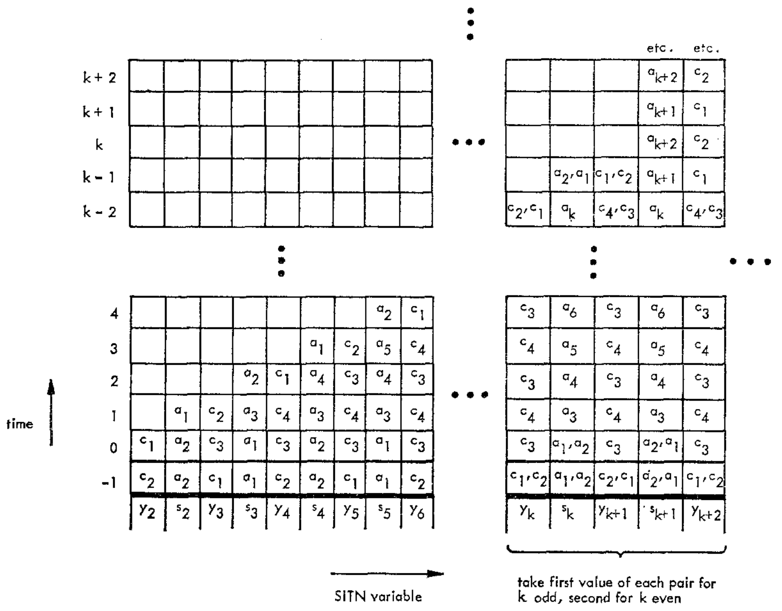

etc.
etc.
k + 2
k + 1
k
k - 1
k - 2
4
3
2
1
0
-1
time
y_2 s_2 y_3 s_3 y_4 s_4 y_5 s_5 y_6
y_k s_k y_{k+1} s_{k+1} y_{k+2}
SITN variable
take first value of each pair for k odd, second for k even

FIG. 4. Successive variable values from $t = -1$ on in the SITN used to prove Theorem 1.

column variable's value at the corresponding row time. Any column left blank above a certain row denotes that the column's topmost entry persists from that row time on. Note that the a_j values assumed by each s_i in Fig. 4 always have (maximum j) $= i$. Hence the $(k + 1)$st cell is the leftmost one which can ever have its s_i variable take on the a_{k+1} value, and from that time on cycle around the a_{k+1}, a_{k+2} loop Since this is the only cycle admitted by Fig. 2, the figure provides the essentials of a proof of Theorem 1.

We fill in the details of our proof in Fig. 5. Figures 5 and 3 clearly indicate that the only possible equilibrium s_i values are a_1 and a_2. Thus it is easily seen that, regardless of the sequence of x_i values along any corresponding SITN in equilibrium, if any single x_i value change is to cause the SITN to enter a cycle, it must do so essentially in accordance with Fig. 4. Hence at least $k + 1$ cells are always required to the right of any single x_i value change if a Fig. 5-type SITN is to enter a cycle under the conditions of Theorem 1. Q.E.D.

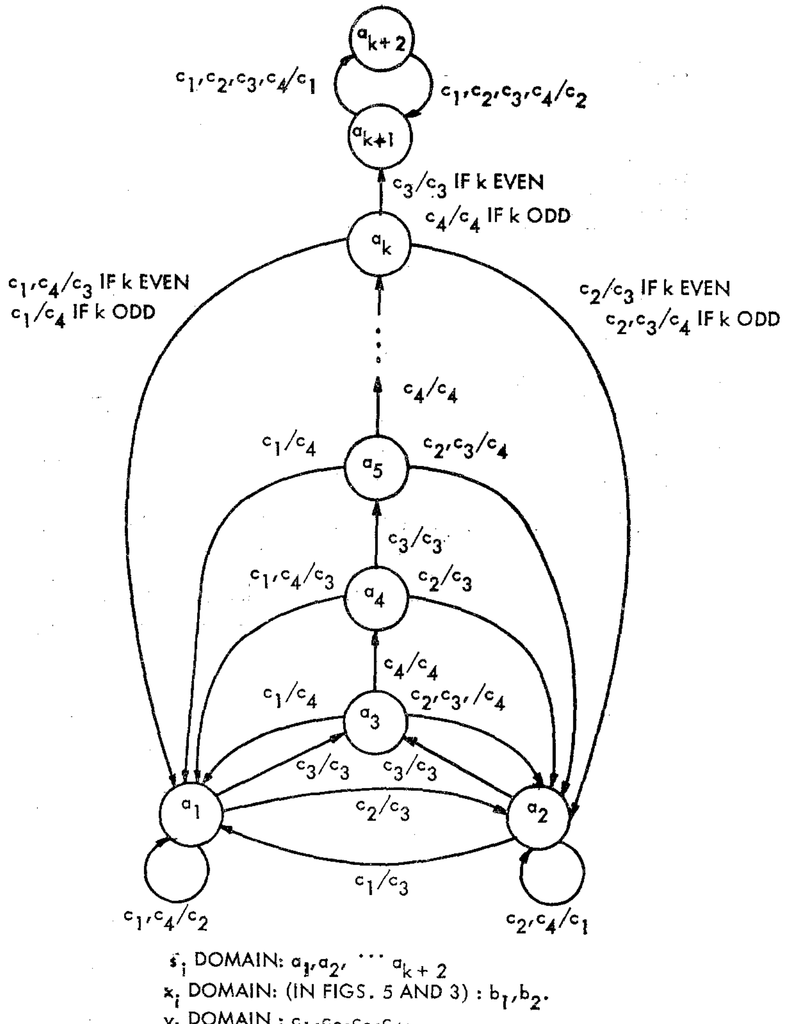


FIG. 5. Complete SITN memory state diagram together with Fig. 3 used in the proof of Theorem 1.

III. SOME UNSOLVABLE PROBLEMS ON CYCLE ENTRY, EQUIVALENCE, AND TRANSIENT CHARACTER

In this section we prove two unsolvability theorems, using essentially one SITN memory state diagram and Minsky's (1961) result that universal Turing machines can be represented by Post tag systems. We state both theorems before proving either.

THEOREM 2. *There does not exist a recursive procedure to determine of an arbitrary SITN cell definition*[2] *whether any corresponding SITN in any arbitrary equilibrium at* $t = -1$ *can have a single* x_i *value change a* $t = 0$ *cause it to:*

(i) *enter a cycle at some* $t \geqq 0$, or

(ii) *pass through a transient which causes a* 1 *output on some* z_i *at some* $t \geqq 0$. The (ii) part of this theorem pertains to the existence of certain SITN equivalence tests (cf. Hennie, 1961).

We now consider SITNs which, if in equilibrium at $t = -1$ and subjected to single x_i value changes at $t = 0$, admit only transient responses (i.e., no cycle entries at any $t \geqq 0$). We call such SITNs *transient SITNs*. Suppose a transient-SITN cell definition is such as to insure that all single x_i changes from equilibrium which cause any y_{i+j} changes at all for $j > 1$, cause y_j value changes all the way to the right boundary of every corresponding SITN. We call such a cell definition *boundary transient*. On the other hand if a transient-SITN cell definition is such as to insure that no single x_i change from equilibrium can cause transients involving y_i value changes more than a bounded (hence calculable) number of cells to the right of the x_i change in any corresponding SITN, we call the cell definition *bounded transient*.

THEOREM 3. *There does not exist a recursive procedure to determine whether an arbitrary transient-SITN cell definition is:*

(i) *boundary transient*, or

(ii) *bounded transient.*

Our first step in proving Theorems 2 and 3 is to define a Post tag system. Let L be a finite set of letters $l_1, l_2, \cdots l_m$; and let W be an associated set of words, such that for each i, W_i is a fixed, finite string or word of letters of L. Let P be some integer, and define the following process applied to some initially given string S of letters of L: Examine the first letter of the string S. If it is l_i, remove the first P letters of S, and then adjoin the word W_i to the right end of the remaining string. Perform the same operations, defined a *production*, on the string that results, and continue making productions so long as there are P or more letters left in each resulting string. If at some point there are fewer than P letters left in a resulting string, the process is said to *terminate* at that string. We call L, W, S, P, and the process just defined a *Post tag system*.

[2] Such as are given in Figs. 3 and 5, for example.

Minsky (1961), showed that the problem of determining for any given Post tag system whether the corresponding process ever terminates is recursively unsolvable.

Before embarking on the remaining details of our proof, we first discuss its main idea. Our plan is to design an SITN cell whose sequential y_i-input to y_{i+1}-output transformation, for x_i constant, directly represents a corresponding production in a given Post tag production process. Our cell thus regards any finite length sequence presented one letter per time unit over its y_i line as a Post tag string viewed one letter per time unit from left to right. The cell's operation requires that it first store in memory the first letter of its y_i input string; then count subsequent input letters until it reaches P; then pass from y_i to y_{i+1} all of the remaining input sequence; and finally append the W_i sequence associated with the first letter it received. Since there are only finitely many finite length W_i strings, this only requires a finite cell. We further arrange our SITN cell so that certain x_i value changes occurring in corresponding SITNs in equilibrium always produce the initial string, S, of our Post tag system on a nearby y_j line. Thus each such x_i change starts a succession of y_i to y_{i+1} sequence transformations that directly represents the corresponding succession of productions in our Post tag system. In order to complete the proofs of Theorems 2 and 3, we augment our cell design so that the various cases of those Theorems either are or are not present according as our Post tag production process does or does not terminate.

We now proceed to the details of our SITN cell specification. We first replace the letters $l_1, l_2, \cdots l_m$ of our given Post tag system by the y_i values $c_1, c_2, \cdots, c_m$, respectively, and then complete the y_i domain by adding three special values, ϕ, ω_1, and ω_2. The latter two are interpreted as marker values, and ϕ is interpreted as the null value. Next we specify a_0, a_1, a_2, a_3, and a_4 as the only possible equilibrium s_i values, and b_1, b_2 as the x_i domain. Figures 6 and 7 then give the main operational part of the SITN memory state diagram that we will use to prove Theorems 2 and 3. Our notation in these figures is as follows: C denotes any y_i value; $\sim\omega_1$, $\sim\omega_2$, and $\sim\phi$ denote any y_i values but ω_1, ω_2, and ϕ, respectively; y_i values raised to the jth power denote j successive repetitions of those values; $l(T)$, for any string of letters T, denotes the number of letters in T; and TU, T, and U both strings, denotes the string consisting of the letters of T followed by the letters of U in order.

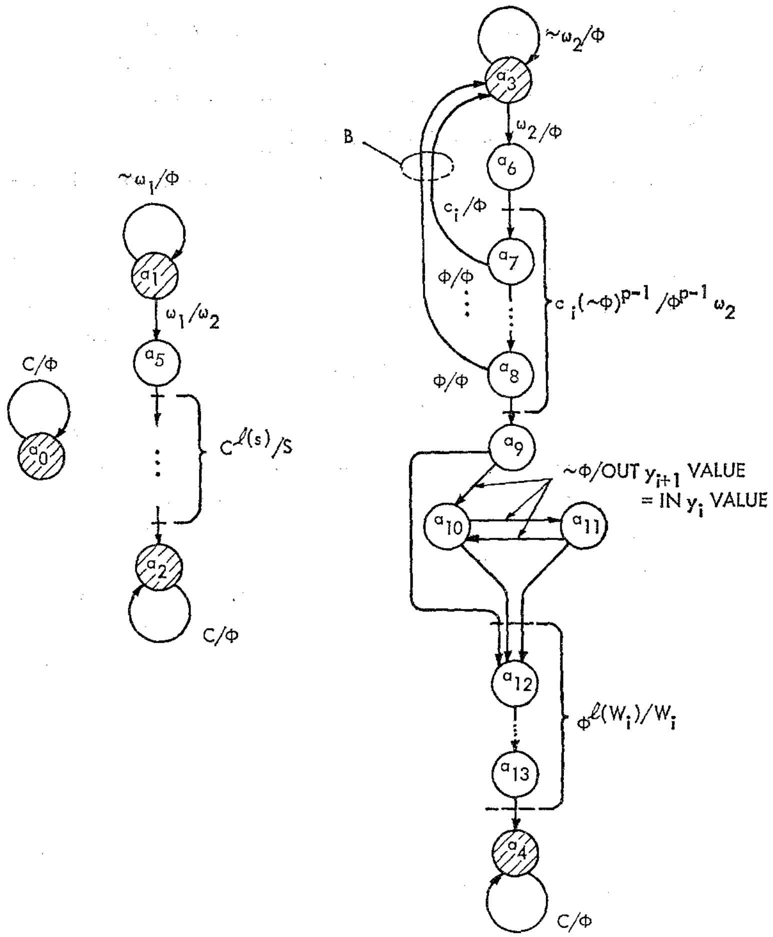
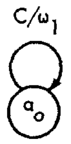


FIG. 6. Partially complete SITN memory state diagram for $x_i = b_1$ used in the proof of Theorems 2 and 3.

Let us now assume, in order to explain Figs. 6 and 7, that we have a corresponding SITN in equilibrium at $t = -1$ as follows: all x_i values are b_1, s_1 is a_0, s_2 is a_1, and all other s_i are a_3. Then suppose that at $t = 0$ there is a single x_1 value change from b_1 to b_2 in the first cell. This causes y_2 to change from ϕ (i.e., null) to ω_1. Subsequently s_2 passes from a_1 down through a_5 to a_2, causing y_3 to put out the string $\omega_2 S$ before settling down at ϕ.



THE REST OF FIG. 7 IS THE SAME AS THE CORRESPONDING PART OF FIG. 6, EXCEPT a_1 IS INTERCHANGED WITH a_2, AND a_3 IS INTERCHANGED WITH a_4.

FIG. 7. SITN memory state diagram for $x_i = b_2$ used in the proof of Theorems 2 and 3.

In other words, the x_1 change causes y_3 to put out essentially the initial string of our given Post tag system. Next we show that y_4 accordingly puts out essentially the result of the first production in the corresponding Post tag system. To see this, we note that s_3 is taken from a_3 to a_6 by $y_3 = \omega_2$; from a_6 to a_7 by the next value of y_3 (i.e., the first letter of S, assumed c_i), and from a_7 to a_9 by the next $P - 1$ values of y_3 (i.e., the next $P - 1$ letters of S, whatever they might be). The value of y_4 remains ϕ, or null, during all of these changes. Following them, however, the $(P + 1)$ non-ϕ value of y_3 (i.e., the Pth letter of S) produces ω_2 out at y_4 . Then the sequence consisting of y_3's $(P + 2)$ non-ϕ value to its last non-ϕ value (i.e., the $(P + 1)$ to the last letter of S) duplicates itself out at y_4 . Finally, when y_3 settles down at ϕ, s_4 leaves a_{10} or a_{11} (whichever state it is in) and causes y_4 to put out the string W_i , corresponding to the first letter of S. After that y_4 also settles down at ϕ. Thus the first production in the Post tag system,

$$S = \underbrace{c_i T_1}_{P \text{ letters}} \underbrace{T_2} \rightarrow \underbrace{T_2 W_i},$$

is represented by the $y_3 \rightarrow y_4$ transformation across the third SITN cell, $\omega_2 S \rightarrow \omega_2 T_2 W_i$ (preceding and succeeding ϕ values not shown).

More generally, the $y_i \rightarrow y_{i+1}$ transformation across the ith SITN cell can be made to represent the $(i - 2)$ Post tag system production as follows: For each i in the tag system alphabet of letters, $\{c_i\}$, we add to Fig. 6 a portion of s_i state diagram which is exactly the c_ith counterpart of the portion already there from a_7 to a_{13} . This is primarily to enable the first c_j value in each incoming y_i sequence to direct s_i to a portion of over-all state diagram that ends y_{i+1}'s non-ϕ sequence with the right W_j . The W_j are produced in one of the a_{12}–a_{13}-type portions of augmented s_i state diagram, and are SITN representations of completions of corresponding tag production steps. The purpose of our wanting to add a_7–a_{11} as well as a_{12}–a_{13}-type portions of s_i state diagram to Fig. 6 is threefold: (*1*) The a_7–a_8 portions are to enable the *i*th cell to effectively remove (i.e., replace by ϕ) the 2nd to Pc_j values of each incoming y_i sequence. This begins the SITN representation of each corresponding tag production. (*2*) If there are ever fewer than Pc_j values, the a_7–a_8 portions are to allow y_{i+1} to remain fixed at ϕ. In each such case one *B* bundle arrow in the augmented Fig. 6 is traversed. (*3*) The a_9–a_{11}-type portions are to enable the *i*th cell to simply pass the $(p + 1)$st-to-last c_j values of each y_i sequence. Thus they represent intermediate tag production steps, preparatory to W_j adjoinments.

Hence if the corresponding tag system productions terminate at the

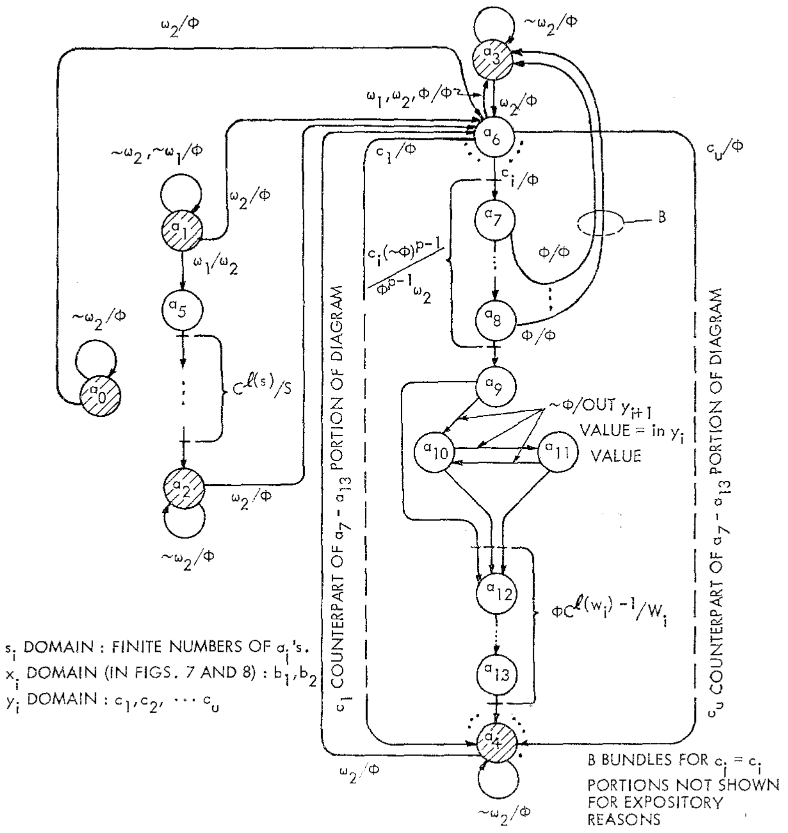

ω2/Φ
~ω2/Φ
ω1,ω2,Φ/Φ
ω2/Φ
~ω2,~ω1/Φ
cu/Φ
c1/Φ
ω2/Φ
ci/Φ
ω1/ω2
B
ci(~Φ)^(p-1) / Φ^(p-1) ω2
Φ/Φ
~ω2/Φ
Φ/Φ
C^ℓ(s)/S
c1 COUNTERPART OF a7 - a13 PORTION OF DIAGRAM
cu COUNTERPART OF a7 - a13 PORTION OF DIAGRAM
~Φ/OUT yi+1 VALUE = in yi VALUE
ω2/Φ
~ω2/Φ
ΦC^ℓ(wi)-1/Wi
si DOMAIN : FINITE NUMBERS OF ai's.
xi DOMAIN (IN FIGS. 7 AND 8) : b1,b2
yi DOMAIN : c1,c2, ··· cu
ω2/Φ
~ω2/Φ
B BUNDLES FOR ci = ci PORTIONS NOT SHOWN FOR EXPOSITORY REASONS

FIG. 8. Complete SITN memory state diagram together with Fig. 3 used in the proof of Theorem 3.

ith string, y_{i+2} is the leftmost SITN value that is left unchanged in the associated network transient.

We now finish our proof of Theorem 3 by filling in the indicated augmentation of Fig. 6 in Fig. 8. We leave it to the reader to check in Figs. 7 and 8 that, following any single x_i perturbation of a corresponding SITN at equilibrium, only one type of transient can involve y_{i+j} value changes for $j > 1$, and that is the type discussed above. Note that in Fig. 8 the ω_2 output arrows from states a_0, a_1, a_2, and a_4 are to insure that every Post tag process representation that gets started in our SITN is carried through to its proper conclusion. Also, the interchange of states alluded to in Fig. 7 is so that the corresponding SITN will not necessarily have to compute to all a_0, a_2, and a_4 "dead" states after many single x_i changes have occurred. Now since the problem of determining whether the tag system productions corresponding to our single transient type ever terminate is recursively unsolvable, so also is the problem of determining whether our corresponding SITN cell is bounded or boundary transient. This completes our proof of Theorem 3.

The proof of Theorem 2 follows almost trivially. We prove part (ii) by modifying the B bundle in Fig. 8 as follows: Instead of directing this bundle into a_3, we direct it into a new state, a_{14}, as shown in Fig. 9. Then we specify that $z_i = 0$ for all s_i values except a_{14}, in which case $z_i = 1$. Theorem 2(ii) follows immediately by noting that it is yes if

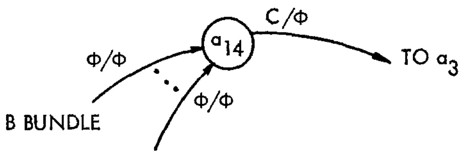


FIG. 9. Change in Fig. 8 for proving Theorem 2ii

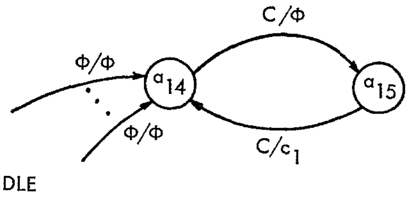


FIG. 10. Modification of Fig. 9 for proving Theorem 2i

and only if the transient in the proof of Theorem 3 is bounded. But this question is recursively unsolvable.

We prove Theorem 2(i) by modifying Fig. 9 as shown in Fig. 10. Then states a_{14} and a_{15} comprise the only cycle that is accessible under the conditions of Theorem 2. Hence Theorem 2(i) is yes if and only if Theorem 2(ii) is yes, which is recursively unsolvable. Q.E.D.

As a passing point, we note that by identifying the right and left boundary signals in the SITNs of Theorem 3, we can get a result much like Theorem 1, but with the inequalities reversed. Although this point has considerable interest, we will not develop it further here.

IV. APPLICATIONS TO NON-SITN MODELS

In this section we apply our results to some non-SITN models discussed in Kilmer (1961, 1962a, b). Our method is to develop a chain of equivalences from one of those models to SITNs.

First, we define the network model shown in Fig. 11. The large square boxes there represent identical combinational logic cells, each having zero switching delay, and the small rectangles represent unit delay elements. Cellular α, β, and x inputs are constant during each unit time interval, so the network operates synchronously. Each cell's α_i and β_i lateral inputs and x_i external input take on values ranging over finite α_i, β_i, and x_i domains, respectively. Correspondingly, each cell's α_i and β_i lateral outputs and z_i external output range over finite domains of values as determined by the truth table comprising the network's cell definition. We require only that the number of network cells be finite, and define such networks *BITNs* (for Bilateral ITerative Networks).

In Fig. 12 we show a reconception of Fig. 11. There the ith cell maps

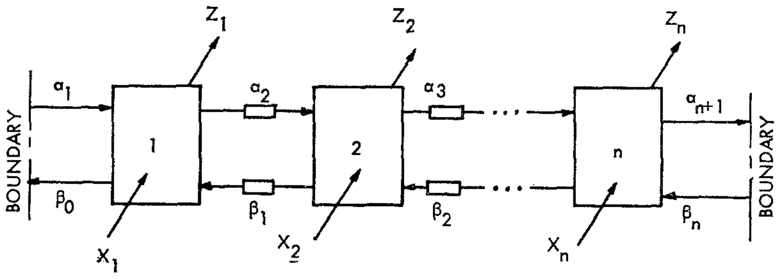


FIG. 11. A BITN

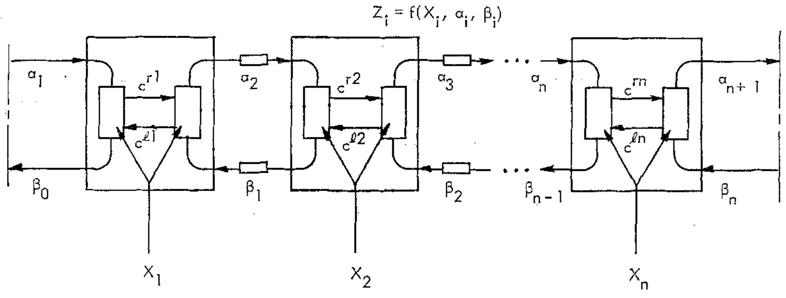


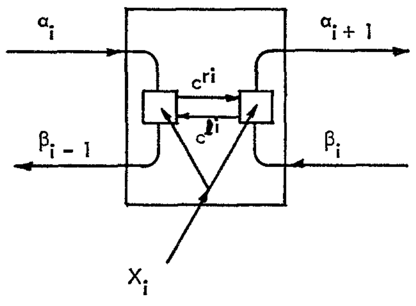
FIG. 12. A reconception of Fig. 11

$\alpha_i \to \beta_{i-1}$ under the influence of x_i and a *left-coupling parameter*, C^{li}; and also maps $\beta_i \to \alpha_{i+1}$ under the influence of x_i and a *right-coupling parameter*, C^{ri}, all with zero delay. The coupling idea is to let C^{li} be a function of β_i such that all those β_i values which exert the same influence in *every* $\alpha_i \to \beta_{i-1}$ mapping cause the same C^{li} value; and similarly for C^{ri} and α_i values.

Figure 11 to Fig. 12 transformations are easily made 1 to 1. To illustrate, assume in Fig. 13 that the $\alpha_i\beta_i\alpha_{i+1}$ portion of the left-hand table is identical for all x_i values. Then β_i maps into α_{i+1} in the same way regardless of x_i's value and whether α_i's value is 0 or 2. Therefore let $C^{ri}(\alpha_i)$ be R_1 for $\alpha_i = 0$ or 2, and R_2 for $\alpha_i = 1$; and similarly for the right-hand table, assume that the $\beta_i\alpha_i\beta_{i-1}$ portion of the table is identical for all x_i values. Then let $C^{li}(\beta_i)$ be L_1 for $\beta_i = 0$, and L_2 for $\beta_i = 1$ or 2. Our method should be clear by now, so we omit the remaining details.

Henceforth we denote Fig. 12 renditions of BITNs, *BITN*s*. And if the C^l domain has only one element, we denote the corresponding networks *R-BITN*s* (for right-coupled BITN*s). Now consider the R-BITN* shown in Fig. 14. We note that each light-dashed rectangle there encloses a structure which closely approximates an SITN cell. We will show that the network in Fig. 14 is, in fact, equivalent to a SITN.

Suppose in Fig. 14 and C^{rl} maps β_1 into α_2 at t. Then this α_2 produces C^{r2}, which maps β_2 into α_3 at $t + 1$. This α_3 in turn produces C^{r3}, which maps β_3 into α_4 at $t + 2$, and so forth. Thus if one knows C^{r1} at t, $t + 2$, $t + 4$, $\cdots$, and one knows β_1 into cell 1 at t, β_2 into cell 2 at $t + 1$, $\cdots$, and β_n into cell n at $t + n - 1$ for a R-BITN*, one has sufficient information to establish exactly half of its α and β values during each



X_i	α_i	β_i	α_{i+1}
0	0	0	1
0	0	1	2
0	0	2	0
0	1	0	1
0	1	1	2
0	1	2	1
0	2	0	1
0	2	1	2
0	2	2	0
1	0	0	1
	⋮		

X_i	β_i	α_i	β_{i-1}
0	0	0	1
0	0	1	1
0	0	2	0
0	1	0	2
0	1	1	2
0	1	2	1
0	2	0	2
0	2	1	2
0	2	2	1
1	0	0	1
	⋮		

FIG. 13. Outline for a Fig. 11 to Fig. 12 transformation

successive time interval. Hence the listed set of β's and associated C^r's is called the R-BITN*'s *correspondence set at t.* (We note that such a set is generally quite distinct from the analogous "initial condition set.")

Obviously any two independent R-BITN*s correspondence sets, say at t and $t + 2k + 1$ for some integer k, respectively, are analyzed separately, yet in the exact same way, in order to determine their respective response. Each set is also analyzed independently of the unit time delay between cells. Hence the R-BITN* in Fig. 14 is *equivalent* to the SITN in Fig. 15 under the conditions that:[3]

(*1*) the small rectangles beneath each cell in Fig. 15 represent unit delays; and

[3] Hennie (1961) has developed a class of equivalence results that are related to, but essentially distinct from, those derived above.

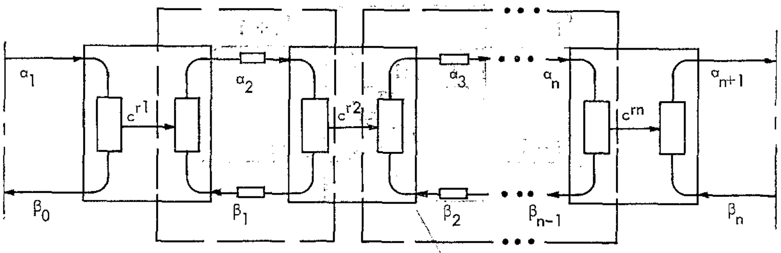


FIG. 14. An R-BITN*

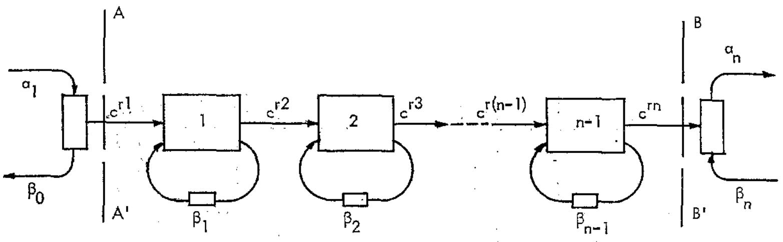


FIG. 15. A SITN equivalent of an R-BITN*

(2) in Fig. 15, C^{ri} and β_i into cell i at time t map into $C^{r(i+1)}$ and β_i out of cell i at time t, just as in Fig. 14, C^{ri} and β_i of the correspondence set at t map into $C^{r(i+1)}$ and β_i of the correspondence sets at t and $t + 2$, respectively.

From this discussion, we readily see Theorem 4.

THEOREM 4. *For each SITN result in Theorems* 1, 2, *and* 3, *there are exactly analogous results for R-BITN*s, BITN*s, and BITNs.* We remark that Theorems 3 and 4, with more or less immediate proof modifications, give strengthened versions[4] of Hennie's (1961) Theorems 10, 10.1, 10.2, 11, 11.1, and 15. Also since the proof of Theorem 3 embodies an SITN representation of a universal Turing machine [cf. Minsky, (1961)], the result clarifies several computing capacity problems alluded to in Hennie (1961). Finally, we claim that the present paper distributes the proof burden for Theorems 3 and 4 in such a manner as to substantially illuminate the basic nature of Hennie's previous work.

V. CONCLUSIONS

Kilmer (1961) and Winograd (1962) essentially closed out the main switching transients problems for BITNs which have either α_i or β_i lines missing. Hennie's previous work (1961) extends these results to canonical decompositions of all BITNs which have cell designs such as

[4] Because we start from equilibrium instead of arbitrary initial conditions.

to prevent any corresponding n-celled network from ever exhibiting over-all memory in the equilibrium state.[5] Kilmer (1962b) discusses the unsolvable nature of steady-state cycling problems in BITNs, BITN*s, and R-BITN*s. And this paper proves the essential unsolvability of the main transients problems in general BITNs and SITNs.

In closing we note the curious duality between Theorem 1 of this paper and Theorem 2 of Kilmer (1962a). The latter states: "For every positive integer k, there exists a BITN cell definition such that every corresponding n-celled BITN is or is not over-all memoryless in the equilibrium state (and hence decomposable into Hennie's canonical form) according as n is or is not $\leqq k$." The interesting thing about this pair of Theorems is that both of their (constructive) proofs seem to require cell complexities that are directly proportional in some sense to k. For Theorem 1 this proportionality is between k and the size of the s_j domain; and for Theorem 2 it is between k and the number of rows in the corresponding cellular truth table definition. Thus general BITN cells seem to admit change-overs of behavioral character in their corresponding n-celled network systems at critical network sizes which are bounded above by a constant times some measure of cell complexity

ACKNOWLEDGMENT

The author wishes to acknowledge several stimulating discussions with Dr. Warren S. McCulloch, of the Research Laboratory of Electronics, Massachusetts Institute of Technology, concerning the work reported in this paper.

RECEIVED: January 31, 1963

REFERENCES

DAVIS, M. (1958). "Computability and Unsolvability." McGraw-Hill, New York.

HENNIE, F. (1961). "Iterative Arrays of Logical Circuits." The M.I.T. Press, Cambridge, Mass., and Wiley, New York.

KILMER, W. (1961). Transient behavior in iterative combinational switching networks. *Proc. AIEE Symp. Switching Theory and Logical Design, September*, pp. 114–128.

KILMER, W. (1962a). Iterative switching networks composed of combinational cells. *IRE Trans. Electronic Computers EC-11*, 123–131.

KILMER, W. (1962b). "On Cycling Behavior in 1-Dimensional Bilateral Iterative Networks." Montana State College Electronics Research Laboratory Report, August.

MINSKY, M. (1961). Recursive unsolvability of Post's problem "tag" and other topics in the theory of turing machines. *Ann. Math.*, *74*, 437–455.

WINOGRAD, S. (1962). Bounded-transient automata. *Proc. AIEE Symp. Switching Theory and Logical Design*, September.

[5] That is, BITNs which have 1 to 1, over-all equilibrium, $\{x_1, x_2, \cdots x_n\}$ input-$\{z_1, z_2, \cdots z_n\}$ output relations.