Identification and Synthesis of Linear Sequential Machines

By P. J. MARINO

(Manuscript received July 14, 1967)

This paper considers the identification and synthesis of linear sequential machines from their state transition tables. Necessary and sufficient conditions for linearity are derived which form the basis of identification tests. A sufficient condition leads to a method for coding the system's state vectors in a fashion consistent with linearity but which does not entail trial and error. The coding process is analytic in nature and allows the coding of state vectors independently of the coding or linearity of the output table. Both the Moore and Mealy models are considered in deriving coding procedures for the input and output vectors.

I. INTRODUCTION

This paper develops a method for identifying and synthesizing linear sequential machines using their state transition table representation. The basic objective is to construct a procedure which can be efficiently implemented by a digital computer. Towards that end, we develop simple and easily used preliminary tests which reject nonlinear systems to precede the time consuming synthesis, or state coding, process. The method for the coding of states is completely analytic, with the result that trial and error processes are not required.

Consider the symbolic state transition table, Table I.

The input vectors, u, have m components ($2 \leqq M \leqq 2^m$),* and the next state vectors, s_{ix}, and present state vectors, s_i, have n unspecified components ($N \leqq 2^n$). The vector components are defined over a modular field, and here this field is taken as $GF(2)$. Most of the results obtained below can be easily extended to other prime fields.

In terms of the state transition table, a linear sequential machine

* This paper considers tables which have at least two distinct columns of next states (nonautonomous systems).

is defined as a system which changes state according to the equation

$$s_{ix} = As_i + Bu_x . \tag{1}$$

A and B are $n \times n$ and $n \times m$ matrices, respectively.* A linear sequential machine is called fully linear when its symbolic output vector, z_{ix}, obeys

$$z_{ix} = Cs_i + Du_x \tag{2}$$

where C and D are matrices of proper size.

When D is the null matrix, the last two equations represent a Moore model of the linear system. Otherwise, the equations describe a Mealy model. Cohn and Even have given a method for model conversion in linear systems.[1]

TABLE I

	u_1	u_2	$\cdots$	u_x	$\cdots$	u_M
s_1	s_{11}	s_{12}	$\cdots$	s_{1x}	$\cdots$	s_{1M}
s_2	s_{21}	s_{22}	$\cdots$	s_{2x}	$\cdots$	s_{2M}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
s_i	s_{i1}	s_{i2}	$\cdots$	s_{ix}	$\cdots$	s_{iM}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$
s_N	s_{N1}	s_{N2}	$\cdots$	s_{Nx}	$\cdots$	s_{NM}

In recent years linear sequential machines have been studied extensively. The motivation for this activity stems from two sources. Not only do linear sequential machines exhibit interesting mathematical and theoretical properties, but they have found a wide range of practical applications; for example, memory addressing circuits, computing over finite fields, counting and timing circuits, error correcting codes, encoding and decoding circuits, and generating pseudorandom and minimum-time test sequences.

As persistent research led to greater understanding, several investigators developed synthesis procedures for linear sequential machines. Davis and Brzozowski[2] have reported a method for the synthesis of nonsingular systems (systems in which, under each input, every pre-

* The addition and multiplication operations are modulo 2. Also, the entries in all matrices are from GF(2).

sent state goes into a unique next state). Their technique is based upon an iterative search over partitions of the system states.

In a mathematically elegant treatment, Cohn and Even[1] have derived a synthesis procedure which is free of trial-and-error processes. Coded output vectors are used to generate the state vector codes. Not only is it necessary that the system have a linear output, but the more severe restriction that the output vectors have been given, *a priori,* a linear coding is also required.

Yau and Wang[3] have disclosed a synthesis technique which does not require a linear and coded output. The construction of the A matrix by examination of a transition graph, which describes the state transitions owing to a given input, leads to the coding of the state vectors. The method requires the system to have 2^n states. When $N < 2^n$, a sufficient number of "don't care" states are introduced to complete the state transition table; however, no suitable procedure is given for the specification or coding of the "don't care" states. The lack of complete freedom from trial-and-error routines is another disadvantage of the method.

In this paper, necessary and sufficient conditions for linearity of the state transition table are derived which lead to the development of the procedure for coding the state vectors. The method accommodates linear systems in general (both singular and nonsingular). The synthesis procedure is analytic and, therefore, no trial-and-error routines are necessary. Also, the state vectors are coded independently of the output table so that the coding process is able to treat systems that have linear or nonlinear, coded or uncoded, output vectors. Both the Moore and Mealy models are considered in deriving coding procedures for the input and output vectors.

II. NECESSARY CONDITIONS FOR LINEARITY

Forming the sum of two next states, say s_{ix} and s_{iy}, under the same present state, s_i, yields

$$s_{ix} + s_{iy} = B(u_x + u_y),$$

since $A(s_i + s_i) = A0 = 0$, mod (2). Since the sum is independent of the present state, it follows that

$$s_{1x} + s_{1y} = s_{2x} + s_{2y} = \cdots = s_{ix} + s_{iy} = \cdots = s_{Nx} + s_{Ny}$$

for each x and y. Let the equality of these sums, for a particular x and y, be denoted by the term state sum, and call the individual sums of pairs of next states component sums. For example, the state sum, $s_{11} + s_{12} = s_{21} + s_{22} = s_{31} + s_{32}$, consists of the component sums $s_{11} + s_{12}$, $s_{21} + s_{22}$ and $s_{31} + s_{32}$.

As a direct consequence of the state sum: if a present state has two identical next states, under two different inputs, then the columns which correspond to the inputs in question are identical, or the table represents a nonlinear machine.*

The state sums of a linear system must be consistent over all pairs of inputs. For example, assume that a state sum contains component sums (written in terms of present state symbols) $s_1 + s_2$ and $s_1 + s_3$. The state sum is consistent only if $s_2 = s_3$; however, if the output table does not allow the reduction of the state transition table by merging s_2 and s_3, then the state sum is inconsistent and the system is nonlinear.

In order to check state sums over the entire table only $M - 1$ state sums are required. Taking the input u_1 as a reference, the state sums

$$s_{11} + s_{1y} = \cdots = s_{i1} + s_{iy} = \cdots = s_{N1} + s_{Ny}$$

for $y = 2, 3, \ldots, M$ cover the table. Since, if $s_{j1} + s_{jy} = s_{i1} + s_{iy}$ for all y of interest, then for any x

$$\begin{aligned} s_{jx} + s_{jy} &= s_{jx} + s_{j1} + s_{jy} + s_{j1} \\ &= s_{ix} + s_{i1} + s_{iy} + s_{i1} \\ &= s_{ix} + s_{iy} . \end{aligned}$$

Therefore, it is not necessary to form state sums for all possible pairs of inputs. For example, consider Table II.

TABLE II

	u_1	u_2	u_3	u_4
s_1	s_2	s_1	s_3	s_4
s_2	s_4	s_3	s_1	s_2
s_3	s_3	s_4	s_2	s_1
s_4	s_1	s_2	s_4	s_3

From inputs u_1 and u_2, $s_1 + s_2 = s_3 + s_4$ (redundant components sums have been deleted) is consistent in the first two columns. From u_1 and

* A similar result has been obtained by Davis and Brozozowski[2] using a different approach.

u_3, $s_2 + s_3 = s_1 + s_4$ and from u_1 and u_4, $s_2 + s_4 = s_1 + s_3$. The last two state sums are rearrangements of the first and therefore, the state sums are consistent over the entire table.

Another symmetry feature appears in singular linear machines (those characterized by a singular A matrix*). If A is singular, then for some present states the rows of next states are identical.[2] This follows since a singular A has rank $r < n$, and therefore, the null space of A has dimension $n - r$, so that $As_i + Bu_x = s_{ix}$ has more than one solution for s_i given s_{ix} and u_2.

The reduced state transition table of a linear sequential machine has additional interesting properties. In what follows only reduced state transition tables are considered unless stated otherwise.

As a preliminary, consider

Theorem 1: If A is nonsingular, then the reduced table of a linear system has an even number of states.

Proof: If A is nonsingular, under each input, each state will appear once, and only once as a next state. Thus, the next state columns are permutations of the present state column. As a consequence, each of the state sums involves all of the system's states. If the number of states, N, were odd, then the same state must appear in two distinct component sums of the same state sum. That is, the state sum contains an equality $s_i + s_j = s_e + s_i$ which implies $s_j = s_e$. But this contradicts the statement that the state table is reduced.

Next, a starting result which connects the number of system states to the number of distinct inputs† is described by

Theorem 2: For a reduced, nonsingular, linear sequential machine which has N states, the number of distinct inputs cannot exceed 2, if $N/2$ is odd, or

$$2 + N \sum_{i=2}^{t} 2^{1-i}$$

where t is the smallest integer for which $N/2^t$ is odd.

Proof: Consider the state sums associated with the first two distinct inputs, u_1 and u_2.

$$s_{11} + s_{12} = s_{21} + s_{22} = \cdots = s_{i1} + s_{i2} = \cdots = s_{N1} + s_{N2}\,. \tag{3}$$

* Common terms from linear and abstract algebra which appear in this paper are treated in several texts; for example, see Birkhoff and MacLane.[4]

† u_x is said to be distinct from u_y if and only if the columns of next states under u_x and u_y are distinct.

Taking u_1 with u_x, a third distinct input, the following state sum is obtained:

$$s_{11} + s_{1x} = s_{21} + s_{2x} = \cdots = s_{i1} + s_{ix} = \cdots = s_{N1} + s_{Nx} \,. \tag{4}$$

Writing segments of equations 3 and 4 in terms of present state symbols as $s_i + s_j = s_e + s_k = s_r + s_u = \ldots$, and $s_i + s_e = s_{j1} + s_{jx} = \ldots$, respectively, leads to the conclusion that state sum consistency requires that equation 4 contain a sum $s_j + s_k$. For if it did not, then locating the terms of equation 4, which contains s_k, say $s_k + s_l$, leads to $s_i + s_e + s_k + s_l = 0$. From equation 3 $s_i + s_j + s_k + s_e = 0$. For compatibility, $s_j = s_l$; so that equation 4 must contain $s_k + s_j$ or contradict the reduction of the table. It then follows that the state sums over distinct pairs of inputs must be mutually derivable via component sums.

Let the component sums obtained from the first state sum from u_1 and u_2 be denoted as follows:

$$S_1 = s_{11} + s_{12} \,, \cdots , S_i = s_{i1} + s_{i2} \,, \cdots , S_N = s_{N1} + s_{N2} \,.$$

Since there are $N/2$ distinct sums, let the symbols $\bar{S}_1 , \bar{S}_2 , \cdots , \bar{S}_{N/2}$ denote the distinct component sums. Then equation 3 can be represented by $\bar{S}_1 = \bar{S}_2 = \cdots = \bar{S}_{N/2}$.

In view of the foregoing, a necessary condition for linearity is that all other state sums must be derived from the sums $\bar{S}_1 , \cdots , \bar{S}_{N/2}$. In generating new state sums the $\bar{S}_i$s are paired and the component sums which are consistent with equation 3 are formed by transposing terms in the resulting equation. For example, pairing $\bar{S}_i$ and $\bar{S}_j$ can yield either of the two equations which do not appear in 3:

$$s_{i1} + s_{j1} = s_{j2} + s_{i2} \,,$$

or

$$s_{i1} + s_{j2} = s_{j1} + s_{i2} \,.$$

Then it is clear that each pairing of the $\bar{S}$s yields two possible state sums. Therefore, the number of unique pairings of the $\bar{S}$s, where each pairing occurs only once (this insures that no component sum will appear in two distinct state sums), is equal to half the maximum number of distinct inputs in excess of the first two.

Separating the $\bar{S}_i$ according to subscript parity gives:

$$\begin{array}{l} \bar{S}_1 \quad \bar{S}_3 \cdots \bar{S}_{N/2-1} \\ \bar{S}_2 \quad \bar{S}_4 \cdots \bar{S}_{N/2} \,. \end{array}$$

If $N/2$ is odd, then one $\bar{S}$ cannot be paired. Therefore, one sum will occur in more than one state sum. Since this is inconsistent with a reduced linear table, a system for which $N/2$ is odd can have only two distinct inputs. If $N/2$ is even, then the number of unique odd to even subscript pairings is $N/4$. The odd-to-odd and even-to-even pairings can be enumerated by considering a single row. It is advantageous to transform the subscripts as follows:

$$S_1' = \bar{S}_2, \qquad S_2' = \bar{S}_4, \cdots, S_i' = \bar{S}_{2i}.$$

Then, separating the new symbols by subscript parity gives:

$$\begin{matrix} S_1' & S_3' \cdots S_{N/4-1}' \\ S_2' & S_4' \cdots S_{N/4}'. \end{matrix}$$

When $N/4$ is odd, no pairing is possible. If $N/4$ is even, the odd-even subscript pairings number $N/8$. Clearly, the odd-to-even pairings of the S' can be treated by reapplying the same transformation to the subscripts. Therefore, the number of allowed pairings is

$$N \sum_{i=2}^{t} 2^{-i},$$

where t is the smallest integer for which $N/2^t$ is odd, if $N/2$ is even. Since each pairing provides for the generation of two distinct columns, in addition to the first two columns, the number of distinct inputs is 2, if $N/2$ is odd or not greater than

$$2 + N \sum_{i=2}^{t} 2^{1-i}, \qquad \text{otherwise.}$$

This completes the proof.*

Theorem 1 leads to a very simple test for the identification of nonlinear tables. The number of states in the table is used to determine the maximum number of distinct inputs. Then, the table is rejected as nonlinear if the number of its distinct inputs, or, equivalently, the number of distinct columns, exceeds the maximum. Table III illustrates the restriction which linearity imposes upon the form of the state transition table.

There are similar, but weaker restrictions associated with the state transition tables of singular linear machines. Consider a system which has N states such that each next state column contains $d(<N)$ distinct states. (The singularity of the A matrix requires that some rows of

* A smaller upper bound can be obtained when $N \neq 2^n$. See the Appendix.

TABLE III

N	Maximum Number of Distinct Columns or Inputs
4	4
6	2
8	8
10	2
12	8
14	2
16	16
70	2
72	56
74	2
76	40
526	2
528	464
530	2
2086	2
2088	1568
2090	2
2092	1048
2094	2
2096	1836

state transition table to be identical.) If the singular matrix A has rank r, then the maximum number of present states which yields the same next state (that is, the maximum number of times a row can be repeated) cannot exceed 2^{n-r}. Since there are $N(\leqq 2^n)$ states, $N2^{r-n} \leqq d \leqq 2^r$. $r \neq n$ implies that each column cannot contain all of the system's states so that the reasoning of the last theorem cannot be applied. In order to gain some insight into how linearity limits the form of the state transition table, consider the case where $N = 2^n$. That is, the set of state vectors form a complete set of n-dimensional vectors with components over $GF(2)$. Let S denote the set of present states, $(s_1, s_2, \cdots, s_N)$, S_{1x} is defined as the set of distinct next states, $(s_{1x}, s_{2x}, \cdots, s_{dx})$ under the input u_x, and it is assumed that $u_1 = 0$. Consider the following

Theorem 3: A linear system which is associated with a singular A matrix of rank r, a null input vector, and which has all states appearing as *next states must have at least 2^{n-r} distinct input vectors, and S_{ix} and S_{iy} are either identical or disjoint for all x and y.*

Proof: First, it will be shown that the S_{1x} are cosets of the group $\{S, +\}$. Since S is a vector space, 0, $s_i + s_j$ are members of S for any s_i, s_j which belong to S. If $u_1 = 0$, then the vectors of S_{11} are a subspace of S because:

(*i*) $A(0) = 0 \in S_{11}$ and
(*ii*) $As_i, As_j \in S_{11}$ implies $A(s_i + s_j) \in S_{11}$

(This follows since $s_i + s_j = s_k \in S$ and $As_k \in S_{11}$.)
The nonull input, u_x, generates cosets of the group $\{S, +\}$, because Bu_x is an n-component vector which must belong to S and therefore, $S_{11} + Bu_x = S_{1x}$.

It is well known that cosets are either disjoint or identical. Since $0 \in S_{11}$, $Bu_x \notin S_{11}$ implies that S_{11} and S_{1x} are disjoint. Therefore, no member of S_{11} can be used as Bu_x if the table is to have a column which contains states not found in column 1. If S_{11} and S_{1x} are to be disjoint, then $Bu_x \in (S - S_{11})$; that is, Bu_x can be selected from a set of $2^n - d$ vectors. Continuing, the next distinct coset is associated with an input, u_y, such that Bu_y has not appeared in any of the preceding cosets. (If it has, then $0 \in S_{11} + Bu_y$.) Then, Bu_y is among $2^n - 2d$ vectors. The last unique cosets is generated from a set of d vectors, or $2^n - kd = d$. So that there are $k + 1 = 2^n/d$ unique cosets of next states in the table. If each present state is to appear as a next state, the table which contains the minimum number of distinct columns must be comprised of one column from each unique coset. Since A has rank r, $d = 2^r$; consequently, there must be 2^{n-r} distinct inputs.

Also, since each unique coset can form 2^r distinct columns, the number of distinct inputs is not greater than 2^n, as expected.

This section has derived several properties that must be exhibited by the state transition table of a linear sequential machine. The consistent state sum requirement will play a central role in the code assignment problem.

III. SYNTHESIS: THE ASSIGNMENT OF LINEAR STATE CODES

To each symbolic state, s, it is necessary to assign a p-dimensional* vector, v, with components over $GF(2)$. That is, $s_{ix} \to v_{ix}$.

The vector assignment must preserve linearity;

$$v_{ix} = Av_i + Bu_x .$$

* $p \geqq n$

Linear systems must give rise to consistent state sums; therefore, the vectors must obey the same sums. Since state sums are only necessary conditions for linearity, a nonlinear state transition function may exhibit consistent state sums.

For any sequential machine the general state transition equation can be written as

$$v_{ix} = Av_i + Bu_x + f_{ix}, \tag{5}$$

where f_{ix} is a p-dimensional vector which is a nonlinear function of the present state and input vectors. When the symbolic states obey the state sums, (equation 3), the vectors must be assigned such that

$$v_{1x} + v_{1y} = v_{2x} + v_{2y} = \cdots = v_{ix} + v_{iy} = \cdots = v_{Nx} + v_{Ny}, \tag{6}$$

for each x and y. Therefore, from equation 5 it is clear that the nonlinear function obeys the same restriction,

$$f_{1x} + f_{1y} = f_{2x} + f_{2y} = \cdots = f_{ix} + f_{iy} = \cdots = f_{Nx} + f_{Ny}. \tag{7}$$

The selection of the A and B matrices exerts some control over the nonlinear function. When $u_1 = 0$,

$$A[v_1 \mid v_2 \mid \cdots \mid v_p] + [f_{11} \mid f_{21} \mid \cdots \mid f_{p1}] = [v_{11} \mid v_{21} \cdots \mid v_{p1}]$$

where

$$[v_1 \mid v_2 \mid \cdots \mid v_p]$$

is a $p \times p$ matrix whose columns are p linearly independent vectors.* Then,

$$f_{11} = f_{21} = \cdots = f_{p1} = 0 \tag{8}$$

can be achieved by

$$A = [v_{11} \mid v_{21} \mid \cdots \mid v_{p1}][v_1 \mid v_2 \mid \cdots \mid v_p]^{-1}$$

Similarly,

$$A[v_1 \mid v_1 \mid \cdots \mid v_1] + B[u_2 \mid u_3 \mid \cdots \mid u_{m+1}]$$
$$+ [f_{12} \mid f_{13} \mid \cdots \mid f_{1,m+1}] = [v_{12} \mid v_{13} \mid \cdots \mid v_{1,m+1}],$$

where $u_2, \ldots, u_{m+1}$ are m linearly independent input vectors, yields

$$f_{12} = f_{13} = \cdots = f_{1,m+1} = 0 \tag{9}$$

* In cases where the system is singular the matrix of next states (the matrix on the right side of the last equality) must be selected so that A has the required rank. The rank of A can be determined directly from the repetition of rows in the transition table.

when

$$B = [v_{12} + v_{11} \mid v_{13} + v_{11} \mid \cdots \mid v_{1,m+1} + v_{11}][u_2 \mid u_3 \mid \cdots \mid u_{m+1}]^{-1}.$$

Additional constraints on the nonlinear function become clear when equation **7** is examined in light of equations **8** and **9**. For $x = 1$, a sample equality in equation **7** is

$$f_{i1} + f_{jy} = f_{j1} + f_{iy}.$$

When $i, j \leqq p$, $f_{j1} = f_{i1} = 0$ (by equation 8); so that $f_{jy} = f_{iy}$. Equation 9 indicates that $f_{jy} = 0$ for $y \leqq m + 1$, $j = 1$. Therefore, f_{iy} vanishes when $i \leqq p$ and $y \leqq m + 1$. Equation 7 implies $f_{ix} = f_{iy} = 0$ and $f_{jx} = f_{jy}$ for $x, y \leqq m + 1$, $i \leqq p <$ j. Finally, $f_{iy} = f_{ix} + f_{jy}$ when $x \leqq m + 1 < y$, and $j \leqq p < i$. Table IV below summarizes the restrictions on the nonlinear function for systems which exhibit consistent state sums.

TABLE IV

	u_1	u_2	$\cdots$	u_{m+1}	u_{m+2}	$\cdots$	u_M
v_1	0	0	$\cdots$	0	$f_{1,m+2}$	$\cdots$	f_{1M}
v_2	0	0	$\cdots$	0	constant	$\cdots$	constant
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\downarrow$		$\downarrow$
v_p	0	0	$\cdots$	0	$f_{1,m+2}$	$\cdots$	f_{1M}
v_{p+1}	$f_{p+1,1}$	$\xrightarrow{\text{constant}}$		$f_{p+1,1}$	$f_{p+1,1} + f_{1,m+2}$	$\cdots$	$f_{p+1,1} + f_{1M}$
$\vdots$	$\vdots$			$\vdots$	$\vdots$		$\vdots$
v_N	$f_{N,1}$	$\xrightarrow{\text{constant}}$		f_{N1}	$f_{N1} + f_{1,m+2}$	$\cdots$	$f_{N1} + f_{1M}$

A particular code assignment can be verified by comparing state transitions along segments of one row and one column with the transitions predicted by the linear equation. If f_{j1} and f_{1x} are found to vanish for $p < j \leqq N$ and $m + 1 < x \leqq M$, respectively, then the code assignment is acceptable. The necessary state transition checks number $M + N - m - p - 1$ (compared with $MN - p - m - 1$ checks if state sum consistency is not verified before a code assignment is attempted). The implication is that sufficiently large values of m and p will force the nonlinear function to vanish over the entire table. While it is undesirable to increase m and p (since this requires more memory elements) it is certainly possible to do so in principle.* However, in computing the A matrix it was convenient to construct a nonsingular matrix of p linearly independent vectors. The state sums (equation 6) which contain no more than $N/2$ component sums imply that at most $N/2 + 1$ states can be assigned vectors independently.

Therefore, not more than $N/2 + 1$ of the coded state vectors are linearly independent with the consequence that it is impossible to form a nonsingular matrix of coded state vectors when $p > N/2 + 1$. Where p exceeds this limit it is possible, in some cases, to express some elements of the A matrix in terms of the remaining elements. This method for finding A is far less attractive than forming a nonsingular matrix of coded state vectors; accordingly, the bound $p \leqq N/2 + 1$ will be enforced. The development which follows demonstrates that the limitation on p does not obscure a system's linearity. Similar considerations lead to $m \leqq M$.

Turning to the state coding problem, the state sum will play an important role in the generation of equations which lead to the linear coding of states. First, attention will be concentrated on nonsingular systems, then a later section will treat singular systems.

3.1 *Nonsingular Systems*

Consider the sum of two component sums

$$v_{ix} + v_{iy} + v_{jx} + v_{jy} = 0;$$

it is true that

$$\begin{aligned} A(v_{ix} + v_{iy} + v_{jx} + v_{jy}) \\ &= v_{ix1} + v_{iy1} + v_{jx1} + v_{jy1} + f_{ix1} + f_{iy1} + f_{jx1} + f_{jy1} \\ &= 0. \end{aligned}$$

(The additional subscript indicates that v_{ix} is the present state which goes into v_{ix1} under the null input, u_1.) Taking v_{ix}, v_{iy}, and v_{jx} among the p independent vectors implies $f_{ix1} = f_{iy1} = f_{jx1} = 0$. Furthermore, assigning vectors such that

$$v_{ix1} + v_{iy1} + v_{jx1} + v_{jy1} = 0 \tag{10}$$

forces f_{jy1} to vanish. The sum (10) must be consistent with the state sums (that is, no state sum can contain an equality of component sums which contradicts equation 10).

* When the input vectors are given as coded it is possible to increase their dimension by a translation.

By similar treatment of all component sums of a state sum, the nonlinear function can be made to vanish in the first column (and therefore, over the first $m + 1$ columns) provided that the attendant increase in the value of p (owing to the designation of independent vectors) does not cause it to exceed its bound.

If all component sums are treated, it is possible to generate an equation of the type 10 in which all four of the vectors have been previously designated linearly independent. Clearly, the equation contradicts the independence of one of the vectors; then, any one of the vectors must be deleted from the set of linearly-independent vectors. The nonlinear function corresponding to the deleted vector can be made to vanish by satisfying the type 10 equation which is obtained when the A matrix operates on the generated equation in question. Consider the following process for treating a single state sum.

(*i*) Select a component sum as a reference sum. Add another component sum to the reference sum.

(*ii*) Operate on the resulting sum with the A matrix to obtain an equation of the type in equation 10. Designate linearly-independent vectors as required and mark the vectors for which the associated nonlinear function has been forced to vanish.

(*iii*) Verify that the equation obtained in step *ii* is consistent with the state sum and the other equations obtained in *ii*. If all vectors in the equation generated in *ii* are linearly-independent, delete one of the vectors from the set of linearly-independent vectors and repeat step *ii* using the generated equation as the sum upon which A operates.

(*iv*) If one of the type 10 equations has three linearly-independent vectors, use it as the sum in repeating step *ii*. Otherwise, add another component sum to the reference sum and repeat step *ii*. Repeat *ii* and *iii* until an inconsistent equation is generated (the system is nonlinear), or until all vectors have been used (the system is linear).

The first time the process passes through step *ii*, three linearly-independent vectors are required; in subsequent passes at most one additional independent vector is needed. After the first pass through the process $N/2 - 2$ unused component sums remain. Therefore, not more than $N/2 + 1$ independent vectors are required for the process, precisely the upper bound on p. If the state vectors were coded using this process, the system would have an undesirably large number of memory elements. Therefore, this process is not an efficient design procedure. However, completion of the process implies linearity of the first $m + 1$ columns with the result that the A and B matrices can be determined. Linearity of the table over the remaining columns is dependent upon the coding of the input vectors.

As previously indicated, linearity over all of the columns can be attained, in the worst case, by increasing the dimension of the input vectors. In this case, $m = M - 1$ would yield a linear system. If it is assumed that the input vectors are uncoded, or can be recoded, then it follows that completion of the process is a sufficient indication that the system is linear. (The problem of coding input vectors is treated in a later section.) The process will be referred to as the maximum memory process.

In order to illustrate the maximum memory process consider the reduced table, Table V.

TABLE V

	0	1
s_1	s_1	s_6
s_2	s_7	s_8
s_3	s_5	s_3
s_4	s_2	s_4
s_5	s_8	s_7
s_6	s_4	s_2
s_7	s_6	s_1
s_8	s_3	s_5

In terms of the coded vectors the state sum is

$$v_1 + v_6 = v_7 + v_8 = v_3 + v_5 = v_2 + v_4 \tag{11}$$

(which is consistent over the table). Using $v_1 + v_6$ as the reference sum,

$$A(v_1 + v_6 + v_7 + v_8) \\ = f_{1,0} + f_{6,0} + f_{7,0} + f_{8,0} + v_1 + v_4 + v_6 + v_3 = 0.$$

Designating v_1, v_6, and v_7 as linearly-independent vectors and satisfying the equation

$$v_1 + v_4 + v_6 + v_3 = 0 \tag{12}$$

yields

$$f_{1,0} = f_{6,0} = f_{7,0} = f_{8,0} = 0.$$

However, vectors v_1, v_6, and v_4 appear in two component sums of equation 11, the first and last; this implies

$$v_1 + v_6 + v_2 + v_4 = 0.$$

Adding this to equation 12 leads to $v_2 = v_3$. This contradicts the reduction of the table; therefore, equation 12 is inconsistent with the state sum. The system is nonlinear.

As pointed out, the maximum memory process is not a suitable vehicle for the economical design of linear sequential machines. The process extracts a limited amount of information from the state transition table. This can be improved by considering all of the unique state sums (that is, from the $M - 1$ state sums by pairing the first input with every other input), and using such equations as 10, which the process generates, to better advantage.

Consider the following procedure:

(*i*) Form the $M - 1$ unique state sums.

(*ii*) Select a reference sum from one of the state sums. Add another component sum (from the same state sum) to the reference sum.

(*iii*) Operate on the sum with the A matrix. Designate linearly-independent vectors as required. Obtain an equation like equation 10 and verify that it is consistent with the state sums. Mark the vectors in all state sums and equations of this type which have been guaranteed a linear state transition under the null input by this step. If all vectors in the equation obtained have been designated linearly independent, delete any one of these vectors from the set of linearly-independent vectors and repeat this step using the equation as the sum upon which A operates.

(*iv*) In the state sums where at least one component sum has had both vectors marked in step *iii*, search for a component sum which has one vector marked or, search over the type 10 equations for one which has three terms marked. If such a component sum or equation is found, use it in repeating step *iii*. (Since three of the vectors make a linear transition, the nonlinear function which is associated with the fourth vector can be made to vanish in step *iii* without designating another linearly-independent vector.) Otherwise,

(*v*) If the sum of the type 10 equation is unique and has two vectors marked, then use it in repeating step *iii*. Otherwise, (*vi*) Form a new sum for use in step *iii* by adding the reference sum to another component sum (from the same state sum). Repeat step *iii*.

The process is repeated until all of the vectors have been marked in step *iii* (the system is linear) or until an inconsistent equation is generated in step *iii* (the system is nonlinear). The coding process for the state vectors is initiated by assigning arbitrary, but linearly independent, vectors to the state vectors so designated by passes through step *iii*. The remaining state vectors are coded using the type 10 equations which were generated in step *iii* in conjunction with the state sums.

The application of this synthesis procedure is more straightforward than its description would indicate. This is best illustrated by an example. Consider Table VI.

TABLE VI

	$\begin{pmatrix}0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\end{pmatrix}$
s_1	s_6	s_4	s_2
s_2	s_1	s_3	s_8
s_3	s_2	s_7	s_6
s_4	s_3	s_1	s_5
s_5	s_4	s_6	s_7
s_6	s_5	s_8	s_3
s_7	s_8	s_5	s_1
s_8	s_7	s_2	s_4
s_9	s_{11}	s_{12}	s_9
s_{10}	s_9	s_{10}	s_{11}
s_{11}	s_{10}	s_9	s_{12}
s_{12}	s_{12}	s_{11}	s_{10}

In terms of the coded vectors, the state sums are: from inputs $\begin{pmatrix}0\\0\end{pmatrix}$ and $\begin{pmatrix}1\\0\end{pmatrix}$,

$$v_6 + v_4 = v_1 + v_3 = v_2 + v_7 = v_5 + v_8 = v_{11} + v_{12} = v_9 + v_{10} \, , \quad (13)$$

from inputs $\begin{pmatrix}0\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$,

$$v_6 + v_2 = v_1 + v_8 = v_3 + v_5 = v_4 + v_7 = v_{11} + v_9 = v_{10} + v_{12} \,. \quad (14)$$

Equations 13 and 14 are mutually consistent, and both are consistent over the table.

The observation that $Av_{12} = v_{12}$ leads to the assignment of the null vector* to v_{12} (therefore, $f_{12,0} = 0$). Then in step *ii* of the synthesis process, take $v_{11} + v_{12}$ of equation 13 as the reference sum, and add it to $v_5 + v_8$ of equation 13. In step *iii* form

$$A(v_{12} + v_{11} + v_5 + v_8)$$
$$= f_{12,0} + f_{11,0} + f_{5,0} + f_{8,0} + v_{12} + v_{10} + v_4 + v_7 \,.$$

Since $f_{12,0} = 0$, taking v_{11} and v_5 among the linearly-independent vectors and the satisfying equation

$$v_{12} + v_{10} + v_4 + v_7 = 0 \quad (15)$$

leads to $f_{11,0} = f_{5,0} = f_{8,0} = 0$. Also, the set of vectors for which the nonlinear function vanishes, denoted by L, is

$$L = \{v_{12}, v_{11}, v_8, v_5\}.$$

Equation 15 is the sum of two component sums of equation 14. Therefore, equation 15 is automatically satisfied.

At this point, component sums do not satisfy the conditions of step *iv*. In step *v* equation 15 is not unique. In step *vi* taking $v_2 + v_7$ with the reference sum yields

$$A(v_{12} + v_{11} + v_2 + v_7)$$
$$= f_{12,0} + f_{11,0} + f_{2,0} + f_{7,0} + v_{12} + v_{10} + v_1 + v_8 \,.$$

Since $f_{12,0} = f_{11,0} = 0$, designate v_2 as a linearly-independent vector. Then,

$$f_{2,0} = f_{7,0} = 0$$

if

$$v_{12} + v_{10} + v_1 + v_8 = 0.$$

The last equation is a rearrangement of terms in state sum (14) and therefore it is automatically satisfied.

* The assignment of the null vector is somewhat arbitrary. It has been shown (Yau and Wang[3]) that the null vector can be assigned to any state which is mapped into itself under the null input.

Updating the set L,

$$L = \{v_{12}, v_{11}, v_8, v_7, v_5, v_2\}.$$

The conditions of steps *iv* and *v* are not satisfied; in step *vi*, adding $v_1 + v_3$ and the reference sum yields

$$A(v_{12} + v_{11} + v_1 + v_3)$$
$$= f_{12,0} + f_{11,0} + f_{1,0} + f_{3,0} + v_{12} + v_{10} + v_6 + v_2 .$$

Since $f_{12,0}$ and $f_{11,0}$ have been shown to vanish, including v_1 among the linearly-independent vectors leads to

$$f_{1,0} = f_{3,0} = 0$$

if

$$v_{12} + v_{10} + v_6 + v_2 = 0.$$

The last equation is the sum of two component sums of equation 14. The set L becomes

$$L = \{v_{12}, v_{11}, v_8, v_7, v_5, v_3, v_2, v_1\}.$$

In equation 14, $v_1 + v_8$, and $v_3 + v_5$ have both vectors marked in step *iii* while $v_6 + v_2$, $v_4 + v_7$, $v_{11} + v_9$ and $v_{10} + v_{12}$ each have one vector marked.

In step *iv*, forming

$$A(v_1 + v_8 + v_6 + v_2) = f_{1,0} + f_{8,0} + f_{6,0} + f_{2,0} + v_6 + v_7 + v_5 + v_1 .$$

Since $f_{1,0}$, $f_{2,0}$, and $f_{8,0}$ have been shown to vanish, $f_{6,0} = 0$ if

$$v_6 + v_7 + v_5 + v_1 = 0. \tag{16}$$

Equation 16 is unique and consistent with the state sums.

$$L = \{v_{12}, v_{11}, v_8, v_7, v_6, v_5, v_3, v_2, v_1\}.$$

Proceeding more quickly, A operating on $v_1 + v_8 + v_4 + v_7$ (from step *iv*) yields $f_{4,0} = 0$ and

$$v_6 + v_7 + v_3 + v_8 = 0. \tag{17}$$

The last equation is unique and consistent with the state sums. Update the set L; let A operate on $v_1 + v_8 + v_{11} + v_9$ (from step *iv*) to obtain $f_{9,0} = 0$ and

$$v_6 + v_7 + v_{10} + v_{11} = 0 \tag{18}$$

Equation 18 is unique and consistent and also has three vectors in the set L. Update the set L.

From step *iv*, $A(v_6 + v_7 + v_{10} + v_{11})$ yields $f_{10,0} = 0$ and $v_5 + v_8 + v_9 + v_{10} = 0$ (which is the sum of two component sums of equation 14). Update the set L. Step *iv* gives $v_{11} + v_{12} + v_9 + v_{10}$ which, when operated on by A, yields

$$f_{10,0} = 0 \qquad \text{and} \qquad v_{10} + v_{12} + v_{11} + v_9 = 0$$

(which is the sum of two component sums). L contains all of the state vectors; therefore, the system can be coded.

There are four linearly-independent vectors ($p = n = 4$). Make the following assignment of the linearly-independent vectors:

$$v_{11} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad v_5 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad v_2 = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad v_1 = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Since $v_{12} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}$, all of the component sums in equation 13 equal

$$v_{11} + v_{12} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}.$$

Then, from equation 13 it follows that

$$v_8 = v_5 + \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\1\\0\\0\end{bmatrix},$$

$$v_7 = v_2 + \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\1\\0\end{bmatrix},$$

and

$$v_3 = v_1 + \begin{bmatrix}1\\0\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}.$$

Similarly, the component sums in equation 14 are each equal to $v_9 + v_{11} = \begin{pmatrix}1\\1\\0\\1\end{pmatrix}$ with the result

$$v_6 = v_2 + \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = \begin{pmatrix}1\\1\\1\\1\end{pmatrix}$$

$$v_4 = v_7 + \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\1\\1\end{pmatrix}$$

$$v_9 = v_{11} + \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\1\end{pmatrix}$$

$$v_{10} = v_{12} + \begin{pmatrix}1\\1\\0\\1\end{pmatrix} = \begin{pmatrix}1\\1\\0\\1\end{pmatrix}.$$

To calculate A consider

$$A[v_{11} \mid v_5 \mid v_2 \mid v_1] = [v_{10} \mid v_4 \mid v_1 \mid v_6]$$

$$A\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix} = A = \begin{bmatrix}1&0&0&1\\1&1&0&1\\0&1&0&1\\1&1&1&1\end{bmatrix}.$$

The matrix B satisfies

$$B\begin{bmatrix}1&0\\0&1\end{bmatrix} = [v_4 + v_6 \mid v_2 + v_6],$$

with the result

$$B = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Since equation 16 and 18 are consistent with the state sums and since they were not used in the coding process, they are redundant. It is easy to verify that the equations are satisfied by the code assignment.

The process produced a code assignment for which the coded state vectors had the minimum number of components. This will not be true for all tables. Considering the way the process extracts information from the table leads to the conclusion that the attainment of the minimum component coding vector depends upon the connectivity of the sequential machine. In a case where the machine is not strongly connected there is a possibility the process will require $p > n$.* (The last example involved a machine which was not strongly connected.) In order to illustrate this consider Table VII.

TABLE VII

	0	1		0	1
s_1	s_6	s_4	s_9	s_{11}	s_{12}
s_2	s_1	s_3	s_{10}	s_9	s_{10}
s_3	s_2	s_7	s_{11}	s_{10}	s_9
s_4	s_3	s_1	s_{12}	s_{12}	s_{11}
s_5	s_4	s_6	s_{13}	s_{14}	s_{15}
s_6	s_5	s_8	s_{14}	s_{15}	s_{14}
s_7	s_8	s_5	s_{15}	s_{13}	s_{16}
s_8	s_7	s_2	s_{16}	s_{16}	s_{13}

The coded vectors obey the state sum:

$$\begin{aligned} v_4 + v_6 &= v_1 + v_3 = v_2 + v_7 = v_5 + v_8 \\ &= v_{11} + v_{12} = v_9 + v_{10} = v_{13} + v_{16} = v_{14} + v_{15}\,. \qquad (19) \end{aligned}$$

* A sequential machine is said to be strongly connected if it is possible to reach any state of the system starting in any initial state.

Taking $v_{16} = 0$ leads to the seection of $v_{13} + v_{16}$ as the reference sum since it is an advantage to exploit the fact that $f_{16,0} = 0$. For brevity, the results obtained from step *iii* of the synthesis process are shown in Table VIII.

TABLE VIII

The Sum A Operates Upon	The Equation Obtained	Additions to The Set L	Additions to Linearly Independent
$v_5 + v_{13} + v_8 + v_{16}$	$v_{16} + v_7 + v_{14} + v_4 = 0$	v_{16}, v_{13}, v_8, v_5	v_5, v_{13}
$v_{16} + v_7 + v_{14} + v_4$	$v_{16} + v_8 + v_3 + v_{15} = 0$	v_{14}, v_7, v_4	v_7, v_{14}
$v_{16} + v_{13} + v_6 + v_4$	$v_{16} + v_{14} + v_5 + v_3 = 0$	v_6	
$v_{16} + v_{13} + v_{14} + v_{15}$	$v_{16} + v_{14} + v_{13} + v_{15} = 0$	v_{15}	
$v_{16} + v_{13} + v_7 + v_2$	$v_{16} + v_{14} + v_1 + v_8 = 0$	v_2	
$v_{16} + v_{14} + v_1 + v_8$	$v_{16} + v_{15} + v_6 + v_7 = 0$	v_1	
$v_{16} + v_{13} + v_1 + v_3$	$v_{16} + v_{14} + v_6 + v_2 = 0$	v_3	

At this point it is observed that v_9, v_{10}, v_{11}, and v_{12} are not in L, and more importantly, it is not possible to involve these vectors in a relationship by application of step *iii* without introducing another linearly-independent vector. By continuing the process it can be demonstrated that the system is linear.

$$A(v_{16} + v_{13} + v_{11} + v_{12}) \quad \text{yields} \quad f_{11,0} = f_{12,0}$$

when

$$v_{16} + v_{14} + v_{10} + v_{12} = 0. \tag{20}$$

Then, $A(v_{16} + v_{14} + v_{10} + v_{12})$ leads to $f_{10,0} = f_{12,0}$

when

$$v_{16} + v_{15} + v_{12} + v_9 = 0. \tag{21}$$

A, operating on the last equation, gives $f_{9,0} = f_{12,0}$

when

$$v_{16} + v_{13} + v_{12} + v_{11} = 0 \qquad \text{(automatically satisfied)}. \tag{22}$$

The system is linear since designating v_{12} as a linearly-independent vector leads to

$$f_{12,0} = f_{9,0} = f_{10,0} = f_{11,0} = 0.$$

It is also of interest that the set L which was obtained by the process can be coded using the four linearly-independent vectors. This will be true in general.

In order to insure realization of the transition table with the least number of memory elements, it is important to develop a means for keeping p at its minimum value, n. First, a general method for the reduction of the number of vector components will be developed. Then, the method will be applied to the problem at hand.

Let the set L_n denote the largest set L generated by the synthesis process using n linearly independent vectors; let $\bar{L}_n$ denote the set of vectors which require additional linearly independent vectors in order to become members of L. The n-component vectors y are members of L_n, and the n-component vectors $\bar{y}$ are in $\bar{L}_n$.

The members of L_n can be coded. Taking the first n vectors of L_n as linearly independent (since order is unimportant), the calculation of A, after the coding, leads to

$$A[y_1 \mid y_2 \mid \cdots \mid y_n] = [y_{10} \mid y_{20} \mid \cdots \mid y_{n0}].$$

This can be simplified by coding the linearly-independent vectors such that $[y_1 \mid y_2 \mid \cdots \mid y_n] = I_n$ (the $n \times n$ identity matrix). Then,

$$A = [y_{10} \mid y_{20} \mid \cdots \mid y_{n0}].$$

Suppose the set L_{n+1} is tentatively formed by designating another linearly-independent vector. It is clear that the vectors in L_{n+1} (and $\bar{L}_{n+1}$) have $n + 1$ components. In order to preserve the coding of L_n take $\begin{pmatrix} y \\ 0 \end{pmatrix} \varepsilon\ L_{n+1}$ where $y\ \varepsilon\ L_n$. That is, the vectors which have been coded over n linearly-independent vectors are increased by one component (which is taken as zero. Members of $\bar{L}_n$ which become members of L_{n+1} will be denoted as $\begin{pmatrix} \bar{y} \\ 1 \end{pmatrix}$. Let $\begin{pmatrix} \bar{y}_{n+1} \\ 1 \end{pmatrix}$ denote the $(n + 1)^{\text{th}}$ linearly-independent vector where $\bar{y}_{n+1}$ is any coded vector not in L_n. The $(n + 1) \times (n + 1)$ matrix $\bar{A}$ is given by

$$\bar{A}\left[\begin{array}{c|c|c|c|c} y_1 & y_2 & \cdots & y_n & \bar{y}_{n+1} \\ 0 & 0 & & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c|c|c|c} y_{10} & y_{20} & \cdots & y_{n0} & \bar{y}_{n+1,0} \\ 0 & 0 & & 0 & 1 \end{array}\right]^{*}.$$

* If $\bar{y}_{n+1}$ and $\bar{y}_{n+1,0}$ are in different sets (L_n or $\bar{L}_n$), then the known A matrix can be used to determine the coding of the vector which is in $\bar{L}_n$.

From the previous coding of L_n, it follows that the left side of the last equation can be written as

$$\bar{A}\left[\begin{array}{c|c} I_n & \bar{y}_{n+1} \\ \hline 0 & 1 \end{array}\right].$$

Therefore, it can be verified that

$$\bar{A} = \left[\begin{array}{c|c} A & A\bar{y}_{n+1} + \bar{y}_{n+1,0} \\ \hline 0 & 1 \end{array}\right]. \tag{23}$$

Observe that if $\bar{A}$ has the form

$$\bar{A} = \left[\begin{array}{c|c} A & 0 \\ \hline 0 & 1 \end{array}\right], \tag{24}$$

then $\bar{A}$ operating on any vector which has the form $\begin{pmatrix} \bar{y} \\ 1 \end{pmatrix}$ will yield a vector of the same form. Similarly, all vectors $\begin{pmatrix} y \\ 0 \end{pmatrix}$ are mapped into another vector where the last component is zero. Since all of the n component vectors y (of L_n) and all n component $\bar{y}$ vectors have different codes, then the last component of the vectors in L_{n+1} can be deleted. Also, the matrix A (in equation 23) is the required matrix.

In view of the foregoing, a code transformation, acting on the coded $\bar{y}$, must be found such that $\bar{A}$ has the form of equation 24. It is well known (for example, Cohn and Even[1]) that the code transformation $y' = Ry$, where R is a nonsingular matrix, cannot alter the linearity of a system. From the state transition equation 1 it is easy to show that this type of code transformation produces a new matrix of the form $R\bar{A}R^{-1}$.

It is required to find an R such that

$$R\bar{A}R^{-1} = \left[\begin{array}{c|c} A & 0 \\ \hline 0 & 1 \end{array}\right]. \tag{25}$$

Comparison of equations 23 and 24 indicates that R must have the form

$$R = \left[\begin{array}{c|c} I_n & T \\ \hline 0 & 1 \end{array}\right]. \tag{26}$$

Using equation 26 in equation 25 leads to the following restriction on the vector T:

$$A\bar{y}_{n+1} + \bar{y}_{n+1,0} = (A + I_n)T. \tag{27}$$

Also, applying the coding transformation yields

$$R\begin{pmatrix} y \\ 0 \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

and

$$R\begin{pmatrix} \bar{y} \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{y} + T \\ 1 \end{pmatrix}.$$

$\bar{y} + T$ cannot be a member of L_n if the $(n + 1)^{\text{th}}$ component is to be deleted.

Therefore, the process of reducing the vector components by one is:

(*i*) Determine the vectors T which satisfy (27).
(*ii*) Of the vectors obtained in i, select one which preserves the identity of $\bar{L}_n$.

Returning to the example, it can be verified that the following is an acceptable code for all of the states except v_9 , v_{10} , v_{11} , and v_{12} :

$$v_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad v_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad v_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \quad v_5 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$v_6 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad v_7 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad v_8 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad v_{13} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$v_{14} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad v_{15} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \text{and} \quad v_{16} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The corresponding A matrix is

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

The set $\bar{L}_4$ is $\{v_9 , v_{10} , v_{11} , v_{12}\}$. Set v_9 equal to any vector not in L_4 , for example

$$v_9 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

From the table $v_{9,0} = v_{11}$. To determine v_{11}, add equations 21 and 22. The result is

$$v_{11} = v_9 + v_{15} + v_{13} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

Then,

$$Av_9 + v_{9,0} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

and

$$A + I_4 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Substituting in equation 27 yields

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} T = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

An acceptable solution is

$$T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

since the new v_9, $\begin{bmatrix}1\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\1\\1\end{bmatrix}$, is not a member of L_4. From equations 20 through 22,

$$v_9 = \begin{bmatrix}1\\0\\1\\1\end{bmatrix}, \quad v_{10} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \quad v_{11} = \begin{bmatrix}1\\0\\1\\0\end{bmatrix}, \quad \text{and} \quad v_{12} = \begin{bmatrix}1\\1\\1\\0\end{bmatrix}.$$

It can be verified that this is an acceptable coding.

3.2 *Singular Systems*

The synthesis procedure which was developed in the last section can be readily extended to treat the coding of singular systems.

Consider a state transition table that has d $(d < N)$ distinct rows. For linear systems, tables of this type have columns of next states comprised of sets of d distinct entries. Therefore, the state sums (with respect to the first column) have the form

$$v_{11} + v_{1x} = \cdots = v_{i1} + v_{ix} = \cdots = v_{d1} + v_{dx}, \qquad \text{for } x = 2, 3, \cdots, M. \tag{28}$$

In general a given table can have columns that are disjoint sets of states. Also, when $2d < N$, these columns can generate state sums such that the only state vectors two state sums may have in common are from the first column. State sums of this type will be called distinct state sums.

An attempt to apply the synthesis algorithm of the last section to a table which has distinct state sums can lead to undesirable results. Since the equations determined in step *iii* of the algorithm can contain only the d distinct vectors of the first column, and since all state vectors must be guaranteed a linear transition to column one, it is clear that in step *iv* at least one linearly-independent vector must be designated for each distinct state sum. As a consequence, a system coding that requires a large number of components may result. In order to avoid this situation it is useful to introduce the concept of independent inputs.

The set of independent inputs is the set of input vectors that spans all of the input vectors. That is, if $u_2, \ldots, u_{m+1}$ are the independent inputs (the set of linearly-independent input vectors), then any other vector, say u_y, can be written as

$$u_y = \sum_{x=2}^{m+1} a_{yx} u_x ,$$

where the a_{yx} are constants which can be arbitrarily assigned when the inputs are uncoded, or they can be determined by inspection of u_y when the inputs are coded.

Recall that $Bu_y = v_{iy} + v_{i1}$, for any i; allowing B to operate on the expansion for u_y yields

$$v_{iy} + v_{i1} = \sum_{x=2}^{m+1} a_{yx}(v_{j(x)x} + v_{j(x)1}), \tag{29}$$

where each $j(x)$ can assume any row index.

The state vector v_{iy} can be forced to make a linear transition under the null input if all vectors in column one and all of the $v_{j(x)}x$ have been guaranteed a linear transition in step *iii*, and if (29) holds. To demonstrate this, use the A matrix to operate on the sum of $v_{iy} + v_{i1}$ and another component sum; then, compare the result to that obtained by repeating the operation after $v_{iy} + v_{i1}$ has been replaced by its expansion. The equations of the type (10) which are generated are required to meet the conditions of step *iii*. Since each distinct state sum not associated with an independent input can be treated in this way, these state sums meet the conditions of step *iv* without the designation of a linearly-independent state vector. Also, since all vectors in column one make linear transitions, all other vectors in the state sum can be treated via step *iv*.

After the independent inputs have been identified (for coded inputs), or designated (for uncoded inputs) the synthesis procedure of the last section* can be applied over the state sums associated with independent inputs. The remaining vectors could be treated by employing equations of the type (29) and entering the synthesis process at step *iv*.

A modified synthesis procedure of this nature would require consideration of the equations of the type (29) in the coding process. The modifications are compatible with nonsingular systems.

The remainder of this section develops an alternative synthesis procedure that is better suited to capitalize on the redundancies found in transition tables of singular systems.

* It is necessary to provide for the selection of another distinct state sum after all component sums of the state sum under consideration have been exhausted. This can be accomplished by selecting the reference sum in step *vi* from another state sum.

When, for example, columns 1 and 2 are disjoint sets of states, (28) contains $2d$ distinct vectors. It is asserted that the $2d$ vectors of this distinct state sum can be coded over the $r + 1$ components (where r is the rank of the A matrix) by a modified form of synthesis procedure of the last section and that the remaining vectors can be coded by a simple relationship. Since only one distinct state sum is to be considered, the information concerning the designation and deletion of linearly-independent vectors (in step *iii*) which is implicit in the other state sums must be incorporated. This is readily accomplished by generating equations of the type (10) over the remaining state sums and equations like (29). Then, the unique equations, which must be compatible with state sums, are used to augment the state sum; that is, these equations, as well as the state sum, are used as state sums in coding over $r + 1$ components. (Notice that the equations in question contain only vectors from column one.)

Partition the vectors such that the upper partition contains $r + 1$ components. That is,

$$v_{ix} = \begin{bmatrix} v'_{ix} \\ \hline v''_{ix} \end{bmatrix}.$$

Then, using the augmenting equations,

$$v'_{11} + v'_{12} = \cdots = v'_{i1} + v'_{i2} = \cdots = v'_{d1} + v'_{d2}$$

and the state sums formed by columns which may appear in the table that are permutations of the first or second columns in the synthesis process, will yield a coding of these vectors and an $(r+1) \times (r+1)$ A matrix. Let A' denote this matrix of rank r. The relations

$$N2^{r-n} \leqq d \leqq 2^r$$

(from Section II) and

$$2^{n-1} < N \leqq 2^n$$

imply

$$2^r < 2d \leqq 2^{r+1} \tag{30}$$

which verifies that $r + 1$ components are required to code the vectors in the first two columns.

Before continuing, it is convenient to separate the present states into sets such that all members of a particular set give rise to identical rows of next states. Let V_j denote the set of present states which are associated with the j^{th} row. If one or more members of the set has been coded over $r + 1$ components, then select one such $r + 1$ component vector as the characteristic vector of the set. Denote it by the symbol cv'_j. If no vector in V_j has been coded over $r + 1$ components, the characteristic vector can be determined from

$$A' cv'_j = v'_{j1} .$$

This last equation has at least one solution for cv'_j since the columns of A' must span all of the v' vectors of the first column. Also, the cv'_j so determined is not a characteristic vector of any other set. Then, taking the system A matrix ($n \times n$) as

$$A = \left[\begin{array}{c|c} A' & 0 \\ \hline 0 & 0 \end{array}\right]$$

will make the linearity of the transitions in the first column dependent only upon the coding over v'. For the vectors which have been coded over v', set the remaining components, the v'', equal to the $n-r-1$ component null vector. (This is necessary, if $u_1 = 0$.) That is,

$$v''_{i1} = v''_{i2} = 0 \qquad \text{for} \qquad i = 1, 2, \cdots , d.$$

Thus, all vectors in the first two columns are completely coded.

The remaining vectors can be coded by the following process. Consider the column of uncoded vectors under an independent input u_y. From the state sum of u_1 and u_y, it follows that

$$v_{jy} = v_{11} + v_{1y} + v_{j1} , \quad \text{for} \quad j = 2, 3, \cdots , d. \tag{31}$$

Since v_{11} and v_{j1} are known, once v_{1y} is coded, the coding of all other vectors in the column is determined by equation 31. v'_{1y} can be set equal to the characteristic vector of the set V in which v_{1y} (as present state) belongs. This will insure that the present state v_{1y} makes a linear transition under input u_1. v''_{1y} can be set equal to any $n - r - 1$ component vector that has not been previously used as a v'' vector. When u_y is not an independent input, v_{1y} can be obtained from equation 29 after the independent inputs have been treated; then equation 31 gives the the coding of the remaining vectors in column y.

To illustrate the process consider Table IX.

TABLE IX

	$\begin{pmatrix}0\\0\\0\end{pmatrix}$	$\begin{pmatrix}1\\0\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\0\end{pmatrix}$	$\begin{pmatrix}0\\1\\1\end{pmatrix}$	$\begin{pmatrix}0\\0\\1\end{pmatrix}$
s_1	s_1	s_3	s_5	s_7	s_9
s_2	s_2	s_4	s_6	s_8	s_{10}
s_3	s_1	s_3	s_5	s_7	s_9
s_4	s_2	s_4	s_6	s_8	s_{10}
s_5	s_1	s_3	s_5	s_7	s_9
s_6	s_2	s_4	s_6	s_8	s_{10}
s_7	s_1	s_3	s_5	s_7	s_9
s_8	s_2	s_4	s_6	s_8	s_{10}
s_9	s_1	s_3	s_5	s_7	s_9
s_{10}	s_2	s_4	s_6	s_8	s_{10}

The state sums are

$$\begin{aligned} v_1 + v_3 &= v_2 + v_4 \,, \\ v_1 + v_5 &= v_2 + v_6 \,, \\ v_1 + v_7 &= v_2 + v_8 \,, \\ v_1 + v_9 &= v_2 + v_{10} \,. \end{aligned}$$

The state sums are consistent over the table. Take the second, third, and fourth inputs as independent. All state sums and equations like (29) generate augmenting equations identical to the null vector. To determine r, note $d = 2$; therefore, from equation 30, $r = 1$. The vectors in the first state sum are coded over two components using the synthesis process. The result is

$$v_1' = \begin{pmatrix}0\\0\end{pmatrix}, \quad v_2' = \begin{pmatrix}1\\0\end{pmatrix}, \quad v_3' = \begin{pmatrix}0\\1\end{pmatrix}, \quad v_4' = \begin{pmatrix}1\\1\end{pmatrix},$$

and

$$A' = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The sets V are:

$$V_1 = \{v_1, v_3, v_5, v_7, v_9\},$$
$$V_2 = \{v_2, v_4, v_6, v_8, v_{10}\}.$$

The characteristic vectors are taken as v_1' and v_2', respectively; that is,

$$cv_1' = \begin{pmatrix}0\\0\end{pmatrix} \quad \text{and} \quad cv_2' = \begin{pmatrix}1\\0\end{pmatrix}.$$

The vectors in the first two columns are fully coded by setting the last two components of each vector to zero:

$$v_1 = \begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \quad v_2 = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad v_3 = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \quad v_4 = \begin{bmatrix}1\\1\\0\\0\end{bmatrix},$$

and

$$A = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}.$$

From the second state sum,

$$v_5 = v_1 + v_2 + v_6 .$$

Take

$$v_6'' = \begin{pmatrix}1\\0\end{pmatrix}.$$

Since $v_6 \in V_2$,

$$v_6' = cv_2' = \begin{pmatrix}1\\0\end{pmatrix}$$

therefore,

$$v_6 = \begin{bmatrix}1\\0\\1\\0\end{bmatrix}$$

and

$$v_5 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}.$$

Similarly, taking

$$v_8'' = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

and observing that

$$v_8' = cv_2' \quad \text{and} \quad v_7 = v_1 + v_2 + v_8$$

lead to

$$v_8 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad v_7 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix},$$

respectively. Since the last input is the sum of the third and fourth inputs, $v_1 + v_9 = v_1 + v_5 + v_1 + v_7$. Then,

$$v_9 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{and} \quad v_{10} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

Finally,

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

IV. CODING THE OUTPUT VECTORS

4.1 *The Mealy Model*

Let z_{ix} denote the symbolic output vector for input u_x and present state s_i. Let the output table contain L distinct vectors; then, z_{ix} is an l-component vector ($L \leqq 2^l$) over $GF(2)$. A k-dimensional vector ($l \leqq k$), w_{ix}, is to be assigned to each symbolic output vector.

If the output is linear,

$$w_{ix} = Cv_i + Du_x . \tag{32}$$

This equation implies that all of the equations obtained in terms of the coded state vectors (state sums and equations derived in the coding process) can be converted directly into relationships involving the coded output vectors. For example, if $v_1+v_2 = v_3+v_4$, then from the output table, for any u_y,

$$w_{1y} + w_{2y} = w_{3y} + w_{4y} .$$

Let the equations obtained in this way be referred to as state derived equations.

From equation 32 it follows (by the same reasoning leading up to equation 6) that

$$w_{1x} + w_{1y} = \cdots = w_{ix} + w_{iy} = \cdots = w_{Nx} + w_{Ny} . \tag{33}$$

Such equalities will be denoted by the term output sum.

Clearly, output sums, and state derived equations must be consistent if the output is linear.

It is known that $N = 2^n$ for a reduced table of a fully linear system.* Therefore, if $N < 2^n$ for a reduced table, the output is nonlinear. On the other hand, where $N = 2^n$, the set of coded state vectors forms a complete set of n component vectors. Then, when $l > n$ and there is a null input, say u_1, it is easy to show that the output vectors in column 1 are a subspace of the space of all output vectors. This suggests that the output vectors in column 1 can be coded over n components setting the remaining $l - n$ components to zero (that is, code over the subspace only).

The upper submatrix of the partitioned C matrix,

$$\left[\frac{C_n}{C_{l-n}}\right] (C_n \text{ is an } n \times n \text{ matrix})$$

can be determined by considering the outputs associated with the independent coded state vectors. Then,

$$C_n[v_1 \mid \cdots \mid v_n] = [w'_{11} \mid \cdots \mid w'_{n1}]$$

where the w' are n component vectors. Recalling that $[v_1 \mid \ldots \mid v_n] = I_n$ (for convenience in the state coding process) leads to

* Cohn and Even[1]

$$C_n = [w'_{11} \mid \cdots \mid w'_{n1}].$$

Since the subspace property of the first column must be preserved, it is true that C_{l-n} is the $n \times l - n$ null matrix. Furthermore, C_n and the code assignment over n components can be determined by designating n independent output vectors in conjunction with the state derived equations and outputs sums.

For all columns which are premutations of the first column, the corresponding Du_x must have zero as components $l - n$ through l (again, to preserve the subspace property). More generally, the entries in the first n components of Du_x generate permutations of the first column, while the entries in the remaining $l - n$ components force the output vector out of the n dimension subspace.*

In order to code the output vectors which are not members of the first column, notice that an output sum which involves such a column and the first column has N distinct sums and $2N$ vectors. N of these vectors have been coded (the members of the first column) and one of these vectors appears in each sum of vectors in the output sum. Therefore, if a linearly-independent vector is designated, by assigning a nonzero entry in $l - n$ components, then all of the remaining $N - 1$ vector codes are determined.

For example, if, in equation 33, $x = 1$, then w_{ix}, for $i = 1, 2, \ldots, N$ are vectors in the subspace (and can be coded), then setting w_{1y} equal to a linearly-independent vector or any vector not in the subspace gives the code assignment for *all* other vectors in column y since $w_{iy} = w_{1y} + w_{1y} + w_{ix}$ for any i. The procedure is basically the same as in coding state vectors of a singular system.

The case where $L \leqq N$ can be treated as an obvious special case of the above.

4.2 *The Moore Model*

In the Moore model of a sequential machine the output is a function of the system's state alone. That is, $D = 0$. It is then obvious from equation 32 that the number of distinct output vectors cannot exceed the number of state vectors. Therefore, the method for coding over the n-dimensional subspace introduced in Section 4.1 can be applied directly, using the state-derived equations.

* This property is the identity or disjointness of sets of vectors which can be rigorously proven by an argument parallel to the proof of theorem 3.

V. CODING THE INPUT VECTORS

5.1 *The Mealy Model*

In the Mealy model of a sequential machine it is possible to have inputs which do not generate distinct columns in the state transition table and yet give rise to distinct output columns. Such cases require an interaction in the determination of the B and D matrices.

Let there be M different inputs, and $K(K \leqq 2^k)$ distinct columns in the state transition table. Take one input as the null input, say $u_1 = 0$. Considering one component sum from each state sum, it follows that

$$Bu_x = v_{11} + v_{1x}$$

for each u_x that generates a distinct column. For convenience number such inputs u_1 through u_K. Similar to the approach of Section 4.1, code the input vectors over the first k components. That is, from

$$B = [B_k \mid B_{m-k}] \qquad \text{(where } B_k \text{ has } k \text{ columns)}$$

select

$$B_k = [v_{11} + v_{12} \mid \cdots \mid v_{11} + v_{1,k+1}][u_2' \mid \cdots \mid u_{k+1}']^{-1}, \tag{34}$$

where the u' are linearly independent vectors formed by the first k components of the input vectors, and where $v_{11} + v_{12}, \ldots, v_{11} + v_{1,k+1}$ are the k vectors that span the set of the K distinct sums of the form $v_{11} + v_{1x}$. Since it is only the first k components of the input vectors that influence the state transitions, it must be true that B_{m-k} is the $n \times (m - k)$ null matrix. The remaining k component input vectors can be obtained by solving the set of linear equations* (in matrix form)

$$B_k u_y' = v_{11} + v_{1y} .$$

At this point k components of all input vectors have been coded.

Considering the output table, the D matrix can be determined from

$$D[u_a \mid \cdots \mid u_r] = [w_{11} + w_{1a} \mid w_{11} + w_{1b} \mid \cdots \mid w_{11} + w_{1r}] \tag{35}$$

where the columns of the matrix on the right side span all sums of the form $w_{11} + w_{1y}$, and where $u_a, \ldots, u_r$ are m inputs which are assigned as linearity independent vectors.

* These linear equations must be consistent since the columns of B_k span all vectors of the form $v_{11} + v_{1y}$. (See equation 34.)

Consider the example in Table X.

TABLE X

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
s_1	s_1/z_1	s_1/z_2	s_1/z_5	s_1/z_6	s_2/z_7	s_2/z_8	s_2/z_9	s_2/z_{10}
s_2	s_3/z_2	s_3/z_1	s_3/z_6	s_3/z_5	s_4/z_8	s_4/z_7	s_4/z_{10}	s_4/z_9
s_3	s_4/z_3	s_4/z_4	s_4/z_{11}	s_4/z_{12}	s_3/z_{13}	s_3/z_{14}	s_3/z_{15}	s_3/z_{16}
s_4	s_2/z_4	s_2/z_3	s_2/z_{12}	s_2/z_{11}	s_1/z_{14}	s_1/z_{13}	s_1/z_{16}	s_1/z_{15}

In order to code the system states, consider u_1 and u_8 which generate the only distinct columns. Take $u_1 = 0$. It can be verified that an acceptable coding is

$$v_1 = \begin{pmatrix}0\\0\end{pmatrix}, \quad v_2 = \begin{pmatrix}0\\1\end{pmatrix}, \quad v_3 = \begin{pmatrix}1\\0\end{pmatrix}, \quad v_4 = \begin{pmatrix}1\\1\end{pmatrix};$$

giving

$$A = \begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}.$$

For the output table, $L = 16$, $l = 4$. Calculating the upper partition of the C matrix, C_2,

$$C_2[v_3 \mid v_2] = [w_3' \mid w_2'],$$

or

$$C_2 = [w_3' \mid w_2'].$$

Take

$$w_3' = \begin{pmatrix}1\\0\end{pmatrix}, \quad w_2' = \begin{pmatrix}0\\1\end{pmatrix};$$

then, $C_2 = I_2$ or

$$C = \begin{bmatrix}1 & 0\\0 & 1\\0 & 0\\0 & 0\end{bmatrix}.$$

From the state sum, $w_4' = \begin{pmatrix}1\\1\end{pmatrix}$, and since $v_1 = 0$, it follows that $w_1' = \begin{pmatrix}0\\0\end{pmatrix}$. Clearly, w_1, w_2, w_3, and w_4 are given by including two additional zero components. In order to code column 3 consider the output sum

$$w_1 + w_5 = w_2 + w_6 = w_3 + w_{11} = w_4 + w_{12}\,.$$

Taking $w_5 = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$ leads to

$$w_6 = w_1 + w_5 + w_2 = \begin{pmatrix}0\\0\\0\\0\end{pmatrix} + \begin{pmatrix}0\\0\\1\\0\end{pmatrix} + \begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\1\\1\\0\end{pmatrix}.$$

Similarly,

$$w_{11} = \begin{pmatrix}1\\0\\1\\0\end{pmatrix}, \qquad w_{12} = \begin{pmatrix}1\\1\\1\\0\end{pmatrix}.$$

By taking

$$w_7 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \quad \text{and} \quad w_9 = \begin{pmatrix}0\\0\\1\\1\end{pmatrix}$$

the remaining output vectors can be coded.

Coding the first component of the input leads to

$$B_1 = \begin{bmatrix}1\\0\end{bmatrix}.$$

With the results

$$B = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 0\end{bmatrix},$$

$$u_1' = u_2' = u_3' = u_4' = 0, \tag{36}$$

and

$$u_5' = u_6' = u_7' = u_8' = 1. \tag{37}$$

The output vectors w_2, w_5, and w_7 span all sums of the form $w_1 + w_x$ (notice $w_1 = 0$) $x = 2, 5, 6, 7, 8, 9, 10$. Then,

$$D[u_2 \mid u_3 \mid u_5] = [w_2 \mid w_5 \mid w_7] = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Considering equations 36 and 37, make a linearly-independent assignment of u_2, u_3, and u_5. Say

$$u_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad u_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad u_5 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix};$$

which leads to

$$D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

To determine u_4, for example, set up the linear equations

$$Du_4 = w_6 \qquad (\text{recall } u_4' = 0)$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} u_4 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix},$$

or

$$u_4 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}.$$

Similarly,

$$u_6 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad u_7 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad u_8 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

5.2 *The Moore Model*

For the Moore model all columns of the state transition table are distinct; therefore, the method for coding over k components of u, introduced in the last section, can be applied directly.

VI. LINEARITY AND INCOMPLETELY SPECIFIED SYSTEMS

This section considers the problem of specifying "don't care" entries in the state transition table in a way consistent with linearity; the results can be extended to the output table.

It is obvious that if an incompletely specified table is to have a linear realization, then the entries in the table must obey the same relationships which were developed in the preceding sections. For example, the table must exhibit consistent state sums, and allow completion of the maximum memory process. The resulting restrictions on the unspecified entries may be used to deduce their appropriate assignments.

Consider Table XI in which the "don't care" next states are denoted by the symbol t.

TABLE XI

	0	1
s_1	t_1	s_2
s_2	s_4	s_6
s_3	s_8	t_4
s_4	s_6	s_4
s_5	s_3	s_1
s_6	s_1	s_3
s_7	t_2	s_8
s_8	t_3	s_5

The state sum is

$$v_2 + t_1 = v_4 + v_6 = t_4 + v_8 = v_1 + v_3 = t_2 + v_8 = t_3 + v_5 . \quad (38)$$

From the third and fifth terms, it follows that $t_4 = t_2$. Forming

$$A(v_1 + v_3 + v_4 + v_6)$$

leads to

$$v_6 + v_1 + v_8 + t_1 = 0. \tag{39}$$

Also,

$$A(v_4 + v_6 + v_2 + t_1)$$

yields

$$v_6 + v_1 + v_4 + \bar{t}_1 = 0,$$

where $\bar{t}_1 = At_1$. Locating v_6, v_1, and v_4 in the state sum leads to the conclusion $\bar{t}_1 = v_3$. The present state which gives s_3 as its next state under zero input is s_5. So that, $t_1 = v_5$. The first and last terms of the state sum imply that $t_3 = v_2$. From $A(v_4 + v_6 + v_8 + t_2)$ obtain

$$v_6 + v_1 + t_3 + \bar{t}_2 = v_6 + v_1 + v_2 + \bar{t}_2 = 0.$$

Adding the last equation to equation 39 gives

$$v_8 + t_1 + v_2 + \bar{t}_2 = v_8 + v_5 + v_2 + \bar{t}_2 = 0.$$

However, equation 38 indicates $t_2 + v_8 = v_2 + v_5$; so that $\bar{t}_2 = t_2$. From the table, $t_2 = v_7$. Table XII shows the fully specified table.

TABLE XII

	0	1
s_1	s_5	s_2
s_2	s_4	s_6
s_3	s_8	s_7
s_4	s_6	s_4
s_5	s_3	s_1
s_6	s_1	s_3
s_7	s_7	s_8
s_8	s_2	s_5

APPENDIX

An Addition to Theorem 2

For N not equal to 2^n there is a set of $2^n - N$ vectors that cannot be used as state vectors. However, when the system is linear operating on any unused vector with the A matrix, and adding Bu_z must produce a vector which is also an unused vector. If this were not the case, there would exist a state vector which would give rise to one of the unused vectors as a next state for some input. It follows that a transition table can be constructed containing only the $2^n - N$ unused vectors. Let this system be called a virtual system. From the foregoing, it is clear that the states of the virtual system must obey the restriction of Theorem 2; that is, the number of distinct inputs cannot exceed

$$2 + (2^n - N) \sum_{i=2}^{t} 2^{1-i},$$

where t is as before. Since the virtual and original systems must remain disjoint, the original system must observe the bound. This is a smaller upper bound than obtained in Theorem 2.

REFERENCES

1. Cohn, M. and Even, S., "Identification and Minimization of Linear Machines," IEEE Trans. Elec. Computers, *EC-14,* No. 3 (June 1965), pp. 367–376.
2. Davis, W. A. and Brzozowski, J. A., "On the Linearity of Sequential Machines," IEEE Trans. Elec. Computers, *EC-15,* No. 1 (February 1966), pp. 21–29.
3. Yau, S. S. and Wang, K. C., "Linearity of Sequential Machines," IEEE Trans. Elec. Computers, *EC-15,* No. 3 (June 1966), pp. 337–354.
4. Birhoff, G. and MacLane, S., *A Survey of Modern Algebra,* 3rd ed., New York: Macmillan, 1965.