The Journal of National Chiao Tung University
Vol. 2, December 1976
pp. 27-48

時序機線型實施之可行性

Linear Realizability of Sequential Machines

吳憲明 Shiang-Ming Wu

Department of Computer Science, N.C.T.U.

(Received August 27, 1976)

ABSTRACT — A New approach is presented to determine whether a completely specified sequential machine can be realized by a linear machine, if state-splitting is permitted. A machine is found to be linearly realizable if we are able to construct a "standard array" from its flow table. The standard array provides the flow table of the linear machine pursued. The construction process involves mainly the interchange between rows and columns in an array and hence is feasible for both hand computation and computer program. This method can be applied to general sequential machines that do not require to be strongly-connected, with exactly two inputs, permutation machine, or autonomous machine. An extended result to deal with incompletely specified case is also provided.

Index terms — Linear sequential machines, linear realization, the largest homomorphic image machines, conjugated machine, standard arrays and state-splitting technique.

I. Introduction

Linear sequential machines have been widely applied to various information-transmission systems such as error correcting and detecting codes, digital communication systems, computer control circuitry and sequence generation, etc. So far, the problem of realizing sequential machines by linear circuits has been extensively studied [1]-[20].

It has been shown that there exists a machine which can not be realized by a linear machine [2] and there also exists a non-linear machine which can be realized by a linear machine [14]. Therefore, whether a non-linear machine is realizable by a larger linear machine becomes the topic of our interest.

Hartmanis and Davis [14] have shown that a strongly connected machine with only two input symbols is linearly realizable if and only if its largest homomorphic image which has no identical next-state rows is a linearly realizable permutation machine. This fact reduces the problem to linear realizability of permutation machines. However, the restrictions of strongly-connectedness and with only two input symbols is not necessary.

Davis [4] has given a procedure for expanding, by state-splitting, a non-

linear permutation machine, which must satisfy a number of conditions such as completely specified, strongly-connected, binary inputs, so that the expanded machine is linear. In this paper, we will propose a new approach for testing whether a completely specified sequential machine can be realized by a linear machine and also finding that linear machine if passing the test. The concept of conjugated machine is created to provide an efficient algorithm which involves mainly array manipulation and hence is programmable. Furthermore, this approach is also extended to deal with incompletely specified sequential machines.

II. Background

Here, we shall list some basic definitions and properties which can be found in most of the previous works and are very important to the development of the results in this paper.

Definition 1. A sequential machine is a quintuple $M=(S,I,O,f,g)$, where S, I, O are, respectively, nonvoid finite sets of states, inputs and outputs; and

$f : S \times I$ into S, the next state function,

$g : S \times I$ into O, the output function for Mealy model,

$g : S$ into O, the output function for Moore model.

If only the next state behavior is considered, then O and g are omitted and the machine becomes $M=(S,I,f)$.

We will always work with synchronous sequential machines. The Moore model is considered as a special case of the Mealy model. Unless otherwise mentioned, the machines will always be deterministic. GF(q) stands for the finite field of q elements, where q is a prime or a power of a prime.

Definition 2. The state behavior of a sequential machine is linear, or, one-to-one linearly realizable over GF(q), if and only if there exists a one-to-one mapping C_S from the state set S into a vector space $\underline{S}$, a one-to-one mapping C_I from the input set I into a vector space $\underline{I}$, and two matrices A and B, with all matrices and spaces over GF(q), such that for any s, $s' \in S$ and $u \in I$ with $f(s,u)=s'$,

$$\underline{s}' = A\,\underline{s} + B\,\underline{u} \tag{1}$$

There is no restriction on the dimension of the vector spaces. The set $\{C_S, C_I, A, B\}$ is called a one-to-one linear realization of M; the dimension of the state space is called the dimension of the realization [3].

Definition 3. A machine $M=(S,I,O,f,g)$ is linear, or one-to-one linearly realizable over GF(q), if and only if there exists a one-to-one linear realization of its state behavior $\{C_S, C_I, A, B\}$ and a one-to-one mapping C_O from the output set O into a vector space $\underline{O}$ and matrices C, D, with all matrices and spaces over GF(q), such that for any $s \in S$, $u \in I$, and $w \in O$, with $g(s,u)=w$,

$$\underline{w} = C\,\underline{s} + D\,\underline{u} \qquad (2)$$

There are no restrictions on the dimensions of the state, input and output spaces. The set $\{C_S, C_I, C_O, A, B, C, D\}$ is called a one-to-one linear realization of M.

Definition 4. A machine $M=(S,I,O,f,g)$ is a homomorphic image of a machine $M'=(S',I',O',f',g')$, if and only if there exist three onto mappings:

$$h_1 : S' \to S, \quad h_2 : I' \to I, \quad h_3 : O' \to O$$

such that

$$h_1[f'(s',u')]=f[h_1(s'),h_2(u')]$$

$$h_3[g'(s',u')]=g[h_1(s'),h_2(u')]$$

for all $s' \varepsilon S'$ and $u' \varepsilon I'$.

If only the next state behavior is considered, then O, O', g, g', h_3 are omitted; and I=I', h_2 is an identity mapping, for the sake of simplicity. A machine M' is said to realize another machine M, if and only if M is a homomorphic image of a submachine of M'.

We are interested in the following class of non-linear (i.e., not one-to-one linearly realizable) machines:

Definition 5. A machine M is one-to-many linearly realizable over GF(q), if and only if there exists a machine M' whose state behavior is one-to-one linearly realizable over GF(q) with the same input set, such that M is a homomorphic image of M'.

If the output has been linearly coded, a sequential machine is not possible to have a one-to-many linear realization, hence only the state behavior will be considered and from now on, we will refer to a sequential machine simply as a 3-tuple M=(S,I,f). If the state behavior of a machine M is one-to-one (or one-to-many) linearly realizable, it will be abbreviated as "M is one-to one (or one-to-many) linearly realizable". Throughout this paper, s_{ij} is the state entry in the ith row and jth column in the state transition flow table. The following theorem can be easily proved by verifying (1) of Definition 2.

THEOREM 1. A sequential machine M is one-to-one linearly realizable, if and only if there exists a one-to-one mapping C_S from the state set S into a vector space S such that, in its coded flow table, the following two canditions are satisfied:

(i) $$\underline{s}_{10}+\underline{s}_{ij}-\underline{s}_{1j}-\underline{s}_{i0}=0 \qquad (3)$$

for any i=1, --- N_S; j=0, --- N_I-1, where N_S, N_I are the number of states and input symbols respectively; and

(ii) for a set of $a_i \varepsilon GF(q)$, with $i=1,---N_S$;

if

$$\sum_{i=1}^{N_S} a_i s_i = 0,$$

then

$$\sum_{i=1}^{N_S} a_i s_{i0} = 0 \qquad (4)$$

III. Linear Realizability and Permutation Machines

The technique of state-splitting has been successfully used to linearly realize the state behavior of some sequential machines which are not one-to-one linearly realizable [4],[14]. Also, it has been shown that it is possible, by state-splitting, to reduce the number of variables needed to linearly realize the state behavior of some machines [5].

First, we will consider the completely specified machines. It will be seen that the linear realizability problem can be reduced to those of permutation machines, and from whose flow table of conjugated form, we will try, by state-splitting if necessary, to form a standard array. If the attempt is successful, which can be decided in a finite number of steps, then the given machine is linearly realizable; otherwise it is not.

It is noted that in this paper, linear realizability includes both one-to-one and one-to-many linear realizability.

The reason that linear realizability problem of sequential machines can be reduced to those of permutation machines is based on Theorem 3 below, which was shown by Hartmanis and Davis [14] under the assumption that the given machine is strongly connected and with exactly two input symbols. However, we will see that this restriction is not necessary.

A sequential machine M is a permutation machine, if and only if every input permutes the set of states, i.e., in its flow table, every next state column contains all states of M.

For a sequential machine $M=(S,I,f)$, let $f(s,\sigma)$ denotes the final state of M, after the input sequence σ is applied to the initial state s.

A partition π on the state set S of a sequential machine $M=(S,I,f)$ is said to have substitution property (SP), if and only if

$$s_1 \overset{\pi}{=} s_2 \text{ implies that } f(s_1,u) \overset{\pi}{=} f(s_2,u),$$

for any $u \varepsilon I$ and $s_1, s_2 \varepsilon S$; where $x \overset{\pi}{=} y$ denotes that x, y belong to the same block of partition π.

Definition 6. Let F be the flow table of a deterministic sequential machine M=(S,I,f) and $\pi=\{b_1,b_2,---,b_p\}$ be a partition on S with substitution property. The merged form of F by π, denoted by F^{π}, is obtained by replacing each present and next state entry s by the block of π to which s belongs, and leaving just one among those identical (present and next states) rows; the input symbols remain unchanged. F^{π} represents the flow table of $M^{\pi}=(B,I,f^{\pi})$, where $B=\{b_1,b_2,---,b_p\}$.

It is obvious that for a deterministic machine M, M^{π} is deterministic also and is a homomorphic image of M. We say that M^{π} is obtained by merging the states of M according to SP-partition π; while M is obtained from M^{π} by state-splitting.

Now we are interested in the following problem: Given a sequential machine M=(S,I,f), how to determine whether it is possible to find a machine M'=(S', I,f') which is one-to-one linearly realizable and an SP-partition π on S' such that M'^{π} is isomorphic to M?

Before reaching the criterion -- Theorem 3, we need the following lemmas:

LEMMA 1. If M=(S,I,f) is a permutation machine and π is an SP-partition on S, then M^{π} is a permutation machine.

Lemma 2 and Theorem 2 were proved by Hartmanis and Davis [14] under the restriction that the given machine is strongly connected and with exactly two inputs. This restriction is found to be redundant as follows:

LEMMA 2. If M'=(S',I,f') is one-to-one linearly realizable and it is not a permutation machine, then M' has at least two identical next-state rows in its flow table.

Proof: If M' is not a permutation machine, then there exist $s_1,s_2\varepsilon S'$ such that $f(s_1,u)=f(s_2,u)$ for some $u\varepsilon I$. Since M' is one-to-one linearly realizable, by (3), we have $f(s_1,u')=f(s_2,u')$ for all $u'\neq u$, $u'\varepsilon I$, which shows that there exist at least two identical next state rows in the flow table of M'.

Q.E.D.

THEOREM 2. If M=(S,I,f) is one-to-many linearly realizable and it is not a permutation machine, then M has at least two identical next-state rows in its flow table.

Proof: If M is one-to-many linearly realizable, then by Definitions 5 and 6, there exists a one-to-one linearly realizable machine M'=(S',I,f') and an SP-partition π on S' such that M'^{π} is isomorphic to M. Since M is not a permutation machine, M'^{π} is not either, which implies that M' is not a permutation machine, by Lemma 1. It follows from Lemma 2 that M' has at least two identical next-state rows in its flow tables.

Since M is not a permutation machine, there exist s_i, $s_j\varepsilon S$ such that $f(s_i,u)=f(s_j,u)$, for some $u\varepsilon I$. Now let us assume that M has no identical next-state rows in its flow table, then $f(s_i,u')\neq f(s_j,u')$, for some $u'\neq u$, $u'\varepsilon I$.

It is easy to see that there is no way, by splitting the states of M to find M' such that there exist two identical next-state rows in the flow table of M'. Since it leads to a contradiction, our assumption is impossible and Theorem 2 is proved.

Q.E.D.

Definition 7. Given a sequential machine $M=(S,I,f)$, let τ_1 be the partition on S, which is obtained by identifying those states with identical next-state rows, that is,

$$s_1 \overset{\tau_1}{=} s_2, \quad \text{if and only if}$$

$$f(s_1,u)=f(s_2,u), \text{ for all } u\varepsilon I.$$

This partition τ_1 has substitution property with respect to M and defines a homomorphic image machine M^{τ_1}. If M^{τ_1} still has identical next-state rows, we can repeat the above process to find the image of image machine, and arrive, in a finite number of steps, at the largest homomorphic image machine M^{τ} which has no identical next-state rows.

The necessary part of the following theorem follows form Theorem 2 and the sufficient part can easily be shown without the restrictions of strongly-connectedness and binary input.

THEOREM 3. A sequential machine M is one-to-many linearly realizable if and only if M^{τ} is one-to-many linearly realizable permutation machine.

For a fixed $u_o \varepsilon I$ and for all $x\varepsilon G$, $u_o^{-1}xu_o$ is called the conjugate of x (under conjugation) by u_o (see Fig. 1). Let G be the group of all permutations.

Definition 8. The conjugated form $M^C=(S,I',f)$ of a permutation machine $M=(S,I,f)$ is obtained by adding a minimum number of $x'\varepsilon G$ to form a larger input set I' such that, for each $u_i\varepsilon I'$, there exists $u_j\varepsilon I'$ satisfying $u_j=u_o^{-1}u_i u_o$. M^C is called the conjugated machine. Note that $I \subseteq I' \subseteq G$.

THEOREM 4. A permutation machine M is one-to-one linearly realizable if and only if its conjugated form M^C is one-to-one linearly realizable, with the same state code and matrix A.

Proof: For M and M^C actually represent the same machine.

Observe that in the coded flow table of a conjugated machine M^C, when any one of equation in (4) is applied to one of (3), the result is included in (3). (see Fig. 2)

Hence in the coded flow table of a conjugated machine M^C, (3) is equivalent to the combination of (3) and (4). This fact and Theorem 1 lead to the following theorem.

THEOREM 5. The conjugated machine M^C is one-to-one linearly realizable if

and only if (3) is satisfied in its coded flow table.

It is evident that an SP-partition π with respect to M is also an SP-partition with respect to M^c, hence the result of merging the states of M according to π and then finding its conjugated form by some input u_o is the same as that of finding the conjugated form of M by u_o first, then merging. Note that the machine M considered is a permutation machine. We conclude as follows:

THEOREM 6. A permutation machine M is one-to-many linearly realizable if

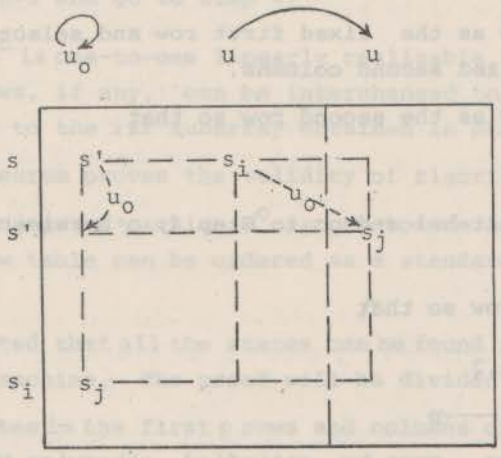


Fig. 1 $u'=u_o^{-1}u\,u_o$ is the conjugate of u by u_o.

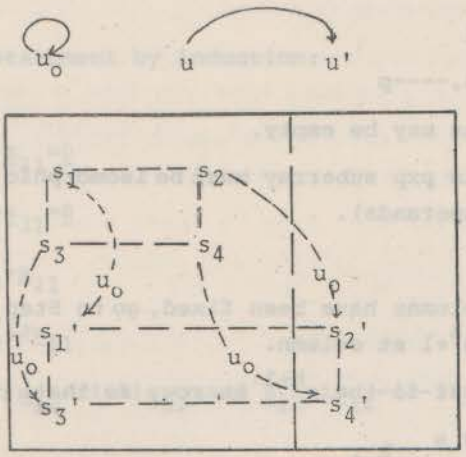


Fig. 2 Conditions (3) and (4) can be merged into (3) for a conjugated machine to be one-to-one line-rly arly realizable.

and only if its conjugated form M^c is one-to-many linearly realizable.

IV. Standard Array

We are going to see that a conjugated machine can be tested for one-to-one linear realizability over $GF(p^m)$ by array manipulation.

The following algorithm is produced to construct a standard array, if possible, by interchanging rows and/or columns in the flow table of a conjugated machine M^C:

Algorithm A1.

(1) Select one row as the fixed first row and select two columns as the fixed first and the fixed second columns.

(2) Select one row as the second row so that

$$s_{21} = s_{12}$$

For p=2, if $s_{22}=s_{11}$, set h=1 and go to Step 4; otherwise M^C is not one-to-one linearly realizable.

For p>2, fix the ith row so that

$$s_{il} = s_{i-1,2} \qquad (5)$$

for i=3,4,----p

At this setp, $s_{p2}=s_{11}$ must be satisfied, otherwise M^C is not one-to-one linearly realizable.

(3) Fix the jth column so that

$$s_{1j} = s_{j1} \qquad (6)$$

for j=3,4,----p

Note that some columns may be empty.

At this step, this pxp subarray must be isomorphic to the modulo p addition table (ignoring the operands).

Set h=1.

(4) If all the columns have been fixed, go to Step 7; otherwise select one column as the fixed p^h+1 st column.

(5) Fix the p^h+1 st to the p^{h+1} st row, so that

$$s_{p^h+i,1} = s_{i,p^h+1} \qquad (7)$$

for i=1,2,---- $p^h(p-1)$

(6) Fix the p^h+2 nd to the p^{h+1} st column, so that

$$s_{1,p^h+i} = s_{p^h+i,1} \qquad (8)$$

for $i=2,3,\text{----}\ p^h(p-1)$

It is noted that some columns may be empty. Let us divide this $p^{h+1}xp^{h+1}$ fixed array A into p subarrays A_i's of p^hxp^h, we get p different subarrays $A_1,A_2,\text{---}$, A_p, all of which are isomorphic to each other. If we consider A as an array consisting of the elements $A_1,A_2,\text{---},A_p$, it is obvious that A is also isomorphic to the modulo p addition table.

Set h = h+1 and go to Step 4.

(7) Set $r=p^h$. M^c is one-to-one linearly realizable, if and only if the remaining non-fixed rows, if any, can be interchanged to make each of the rxr subarrays isomorphic to the rxr subarray obtained in Step 6.

The following theorem proves the validity of algorithm A1.

THEOREM 7. A conjugated machine M^c is one-to-one linearly realizable, if and only if its flow table can be ordered as a standard array (some columns may be empty).

Proof: It is noted that all the states can be found in any column, because M^c is a permutation machine. The proof will be divided into four parts.

(i) All the states in the first p rows and columns of a standard array can be expressed as

$$k\,\underline{s}_{12}-(k-1)\underline{s}_{11} \tag{9}$$

where k GF(p) and $1\leq k\leq p$.

We will prove this statement by induction:
by (3) and (5),

$$\underline{s}_{11}+\underline{s}_{22}-\underline{s}_{12}-\underline{s}_{21}=\underline{0}$$

$$\underline{s}_{21}+\underline{s}_{32}-\underline{s}_{22}-\underline{s}_{31}=\underline{0}$$

$$\underline{s}_{32}=2\underline{s}_{22}-\underline{s}_{21}$$

$$=3\underline{s}_{12}-2\underline{s}_{11}$$

Suppose that $\underline{s}_{i-1,2}=(i-1)\underline{s}_{12}-(i-2)\underline{s}_{11}$ and $\underline{s}_{i2}=i\underline{s}_{12}-(i-1)\underline{s}_{11}$ hold,

$$\underline{s}_{i1}+\underline{s}_{i+1,2}-\underline{s}_{i2}-\underline{s}_{i+1,1}=\underline{0}$$

then

$$\underline{s}_{i+1,2}=2\underline{s}_{i2}-\underline{s}_{i1}$$

$$=2[i\underline{s}_{12}-(i-1)\underline{s}_{11}]-\underline{s}_{i-1,2}$$

$$=(i+1)\underline{s}_{12}-i\underline{s}_{11}$$

Hence,

$$\underline{s}_{k2}=k\underline{s}_{12}-(k-1)\underline{s}_{11} \qquad (10)$$

for k=1,2,---,p

$$\underline{s}_{p2}=p\underline{s}_{12}-(p-1)\underline{s}_{11}=\underline{s}_{11}$$

It follows from (5), (10) that all the first p states in the first and the second columns can be written as (9). These states can never be found in the remaining entries of the first and the second columns, because $\dot{M}^C$ is a permutation machine.

Let us observe that if the first p entries of the jth and the first columns have one state in common, then they have all the first p states in common: The assumption is satisfied by (6) in Step 3, and for i=2,---,p; j=3,---,p,

$$\underline{s}_{11}+\underline{s}_{ij}-\underline{s}_{i1}-\underline{s}_{1j}=\underline{0}$$

$$\underline{s}_{ij}=\underline{s}_{i1}+\underline{s}_{1j}-\underline{s}_{11}$$

$$=[k_1\underline{s}_{12}-(k_1-1)\underline{s}_{11}]$$

$$+[k_2\underline{s}_{12}-(k_2-1)\underline{s}_{11}]-\underline{s}_{11}$$

$$=(k_1+k_2)\underline{s}_{12}-(k_1+k_2-1)\underline{s}_{11}$$

$$=k'\underline{s}_{12}-(k'-1)\underline{s}_{11}$$

where k_1, k_2 GF(p), and $k'=k_1+k_2\varepsilon GF(p)$.

Hence (i) is proved and due to permutation columns, we know that for the first p rows, the first p columns have the same p states as the first column; and for neither i<p<j nor j<p<i can s_{ij} be expressed as the form (9)

(ii) All the states in the first p^2 rows and columns of a standard array can be expressed as

$$k_o\underline{s}_{12}-(k_o-1)\underline{s}_{11}+k_1(\underline{s}_{1,p+1}-\underline{s}_{11}) \qquad (11)$$

For $i,j\leq p$, s_{ij} can be written as (11), with $k_1=0$; $s_{1,p+1}$ can also, with $k_o=0$ $k_1=1$;

for $1<i\leq p$,

$$\underline{s}_{i,p+1}=\underline{s}_{i1}+\underline{s}_{1,p+1}-\underline{s}_{11},$$

with $k_1=1$;

$$\underline{s}_{p+1,p+1}=\underline{s}_{p+1,1}+\underline{s}_{1,p+1}-\underline{s}_{11}$$

by (7) $$=2\underline{s}_{1,p+1}-\underline{s}_{11}$$

$$=\underline{s}_{11}+2(\underline{s}_{1,p+1}-\underline{s}_{11})$$

with $k_o=0$, $k_1=2$;

for $1 \le i \le p^2-2p$,

$$\underline{s}_{p+i,p+1} = \underline{s}_{p+1,1} + \underline{s}_{i,p+1} - \underline{s}_{i1}$$

by (7) $$= 2\underline{s}_{i,p+1} - \underline{s}_{i1}$$

$$= 2(\underline{s}_{i1} + \underline{s}_{1,p+1} - \underline{s}_{11}) - \underline{s}_{i1}$$

$$= \underline{s}_{i1} + 2(\underline{s}_{1,p+1} - \underline{s}_{11})$$

Hence, for $i=1,2,\cdots,p(p-1)$, $s_{i,p+1}$ can be written as (11), and $s_{p+i,1}$ can also, because of (7). Now we can see that all the first p^2 entries of the first column can be written as (11).

Similar to (i), for $j=p+1,\cdots,p^2$, the first p^2 entries of the jth and the first columns have one state in common due to (7) and (8), it follows that they have all the first p^2 states in common.

Hence (ii) is proved and it can be similarly concluded that for the first p^2 rows, the first p^2 columns have the same p^2 states as the first column; for neither $i<p^2<j$ nor $j<p^2<i$ can $\underline{s}_{ij}$ be in the form (11).

(iii) All the states in the first p^h rows and columns of a standard array can be expressed as

$$k_o\underline{s}_{12} - (k_o-1)\underline{s}_{11} + \sum_{i=1}^{h-1} k_i(\underline{s}_{1,p^i+1} - \underline{s}_{11}) \qquad (12)$$

where $h \ge 2$; $k_o, k_i \in GF(p)$

Let us prove it by induction on h: if for $h=h_o$, (12) holds, then we can show that for $h=h_o+1$, (12) still holds, by arguments similar to those given in (ii).

Now it is seen that for the first p^h rows, the first p^h columns have the same p^h states as the first column; for neither $i<p^h<j$ nor $j<p^h<i$ can $\underline{s}_{ij}$ be in the form (12).

(iv) In Step 7 of the procedures to construct a standard array, let n be the number of rxr subarrays. If $n>1$, then all the remaining states other than the first rxr subarray can be expressed in terms of the upper left entry of the local rxr subarray and those entries in the first rxr subarray as the following form:

$$\underline{s}_{(n-1)r+1,j} = \underline{s}_{(n-1)r+1,1} + k_o(\underline{s}_{12} - \underline{s}_{11}) + \sum_{i=1}^{h-1} k_i(\underline{s}_{1,p^i+1} - \underline{s}_{11}) \qquad (13)$$

where $k_o, k_i \in GF(p)$.

(13) is obtained from the following two equations:

$$\underline{s}_{(n-1)r+1,j}+\underline{s}_{11}-\underline{s}_{(n-1)r+1,1}-\underline{s}_{1j}=\underline{0}$$

$$\underline{s}_{1j}=k_o\underline{s}_{12}-(k_o-1)\underline{s}_{11}+\sum_{i=1}^{h-1} k_i(\underline{s}_{p^i+1,1}-\underline{s}_{11})$$

Since each rxr subarray is isomorphic to the first one, all the states can be written as (13) and there must be exactly $nr=np^h$ states and rows, if and only if M^C is one-to-one linearly realizable. The proof for Theorem 7 is now completed.

Q.E.D.

V. State-Splitting Technique

Consider a permutation machine $M=(S,I,f)$, let γ be the smallest SP-partition on S, i.e., $f(s,u_i)\overset{\gamma}{=}f(s,u_j)$, for all $s \in S$; $u_i,u_j \in I$. The corresponding partition on the rows of its flow table is defined as the partition ρ, i.e., row $(s_1)\overset{\rho}{=}$row (s_2), if and only if $f(s_1,u_i)\overset{\gamma}{=}f(s_2,u_j)$, for some $u_i,u_j \in I$; where row (s_i) denotes the row with the present state s_i.

It is evident that both M and its conjugated form M^C have the same γ and ρ. Let $M_1^{\tau C}$ represent a one-to-one linearly realizable machine obtained by state-splitting from $M^{\tau C}$ which is the conjugated form of M^τ. Let R_i, R_i' stand for the subarrays of $M^{\tau C}$, $M_1^{\tau C}$ respectively, associated with one block of ρ; and R_i^o is the subarray obtained from R_i after eliminating repeated columns, if any. We observe the following theorem.

THEOREM 8. $M^{\tau C}$ is one-to-many linearly realizable, if and only if R_i^o, for each block of ρ, can be ordered as a standard array.

The following algorithm is produced for state-splitting $M^{\tau C}$ to find $M_1^{\tau C}$:

Algorithm A2.

Case (i): R_i has repeated columns but not all R_i's have their repeated columns in the same position.

(1) Let v denote the largest multiplicity of the state entries in R_i and $m=\lceil \log_p v \rceil$, where $\lceil x \rceil$ stands for the smallest integer larger than x. Split each state of $M^{\tau C}$ inot p^m states in $M_1^{\tau C}$, and complete the next state flow table for $M_1^{\tau C}$ by the following steps.

(2) Construct the column with input u_o under which M^τ is conjugated into $M^{\tau C}$:

If $f(s_i,u_o)=s_j$ in $M^{\tau C}$,

then $f(s_{i_k},u_o)=s_{j_k}$ in $M_1^{\tau C}$, for $k=1,2,---,p^m$.

(3) Construct R_1': Let v_i denote the multiplicity in R_1 of s_i, for each s_i

$\in R_1^o$. Substitute each s_i in R_1^o by the first v_i columns of a $p^m x p^m$ standard subarray which consists of s_{i_1}, s_{i_2}, ---, $s_{i_{p^m}}$, and leave the remaining $p^m - v_i$ columns empty. The subarray thus formed is R_1'.

(4) Construct R_2', whose next states under u_o are the present states of R_1': From R_1' and the input conjugate relations which are the same for both M^{TC} and M_1^{TC} and M_1^{TC}, R_2' can be obtained.

(5) R_3', R_4', ---, R_t' can be obtained by repeating the same process as Step 4, where t is the number of blocks of ρ. The flow table for M_1^{TC} is completed, if the rows in each R_i', for i=2,---,t, can be interchanged to form a standard array; otherwise, further state-splitting is needed, see Case (ii).

Case (ii): R_i has no repeated columns, but some columns in M^{TC} must be splitted in order to form a standard array, such columns are said to be the conflicting columns.

(1) Split each state s_i of M^{TC} into two states s_i, s_1' in M_1^{TC}.

(2) For each R_i, i=1,---,t, put all the conflicting columns on one side, and duplicate this new array below itself to get R_1' which consists of two identical halves.

(3) Let all the entries of both the conflicting columns in the upper half of R_i' and the non-conflicting columns in the lower half of R_i' be the primed states s_i'; while all the other entries remain unprimed states s_i.

Such a process provides for each R_i a spare block which can be adjusted by interchanging its rows according to the conflicting columns, and thus at least one conflicting column can be handled. That is the reason why we have less conflicting columns after this state-splitting process.

(4) If confliction still remains, repeat the process in Step 3 until a standard array can be formed.

(5) The present-state column for the flow table of M_1^{TC} can be completed as follows:

Let $u_o(s_1, s_2, ---, s_k)$ represent a cycle set under the input u_o:

$$s_1 \xrightarrow{u_o} s_2 \xrightarrow{u_o} --- \xrightarrow{u_o} s_k \xrightarrow{u_o} s_1.$$

If M^{TC} contains $u_o(s_1, s_2, ---, s_k)$, then either

(i) $u_o(s_1, s_2, ---, s_k)$ and $u_o(s_1', s_2', ---, s_k')$; or

(ii) $u_o(s_1, s_2, ---, s_k, s_1', s_2', ---, s_k')$

can be used, along with the conjugate relation of M^{TC}, to determine the present-state column in the flow table of M_1^{TC}.

The following algorithm summarizes the overall process to determine whether a given sequential machine M can be linearly realized over GF(q).

Algorithm A.

(1) Find M^{τ}, the largest homomorphic image of M which has no identical next-state rows in its flow table.

(2) If M^{τ} is a permutation machine, go to Step 3; otherwise, M is not linearly realizable.

(3) Find $M^{\tau c}$, the conjugated form of M^{τ}. If the flow table of $M^{\tau c}$ can be ordered as a standard array, M is one-to-one linearly realizable; otherwise, determine γ, ρ, R_i, R_i^o and go to Step 4.

(4) If each R_i^o can be ordered as a standard array, go to Step 5: otherwise, M is not linearly realizable.

(5) If R_i has repeated columns but not all R_i's have them in the same position -- Case (i) state-splitting; If R_i has no repeated columns but there exist conflicting columns, when trying to form a standard array -- Case (ii) state-splitting. Follow the state-splitting procedure to obtain the flow table for $M_1^{\tau c}$, which is one-to-one linearly realizable and realizes $M^{\tau c}$

(6) From $M_1^{\tau c}$, find M_1^{τ}; and then from M_1^{τ}, find M_1 which is one-to-one linearly realizable and realizes M, thus we obtain a one-to-many linear realization for M.

Two examples are given in Appendix to demonstrate the above algorithm.

VI. Linear Realizability for Incompletely Specified Machines

We will apply the previous results to the case of incompletely specified sequential machine M. The method which we have just discussed can be slightly modified to see whether it is possible to specify all the don't-care entries so that M is linearly realizable.

Definition 9. A sequential machine is incompletely specified, if and only if some next-state entries in its flow table are unspecified, i.e., some of its next-state entries are "don't-care".

Definition 10. For s_i, $s_j \varepsilon S$, s_i and s_j are said to have A-identical (next-state) rows, if and only if there exists some $u \varepsilon I$, such that $f(s_i,u)=f(s_j,u)$, and for all $u' \neq u$, $u' \varepsilon I$, either or both of $f(s_i,u')$ and $f(s_j,u')$ is (are) unspecified, or $f(s_i,u')=f(s_j,u')$.

Definition 11. For s_i, $s_j \varepsilon S$, s_i and s_j are said to have B-identical (next-state) rows, if and only if for all $u \varepsilon I$, there exists no $f(s_i,u)=f(s_j,u)$; either or both of $f(s_i,u)$ and $f(s_j,u)$ is (are) unspecified, or $f(s_i,u)=f(s_j,u)$.

Among the elements in a set S, a relation ~ is said to be transitive, if and only if for $a,b,c \varepsilon S$, $a \sim b$, $b \sim c$ implies $a \sim c$. It is evident that a set S can not be partitioned by a relation without transitivity and neither A-identical nor B-identical defined above possesses transitivity.

The following algorithm is produced to find linear realizability for incompletely specified maching M=(S,I,f).

Algorithm B.

(1) Find τ_1, M^{τ_1};---;τ, M^{τ}, which are similarly defined as those of completely specified machines (see Definition 7) except that A-identical rows with transitivity are partitioned instead of identical rows. If there exist A-identical rows without transitivity, then M is not linearly realizable. Specify the don't-care entries so that the A-identical rows with transitivity become identical, whenever a partition τ_i is obtained.

(2) It is noted that M^{τ} has no A-identical rows. For each set S_i of B-identical rows in M^{τ}, it can be inspected to see whether a further partition τ_o is needed, by considering some subset of S_i as a block of τ_o, so that $(M^{\tau})^{\tau_o}$ can be a permutation machine, when all the remaining don't-care entries are properly specified.

If it is not possible for M^{τ} or $(M^{\tau})^{\tau_o}$ to be a permutation machine, then M is not linearly realizable (This can be decided in a finite number of steps, because the number of possible τ_o is finite too); otherwise go to Step 3.

(3) If the flow table of the conjugated form of the permutation machine obtained in Step 2 can be ordered as a standard array, then go to Step 4; otherwise M is not linearly realizable.

(4) M is linearly realizable and since all the state entries have been already specified, go to Step 3 of algorithm A.

The rule for specification in Step 1 is justified as follows: a machine M can not have a next-state configuration like

	u_1	u_2
s_i	a	b
s_j	a	c

with $b \neq c$,

if M is linearly realizable (ref. Lemma 2 and Theorem 2).

VII. Discussion

The method for linear realization which we have discussed here does not require the given machine to have the following restrictions:

(i) strongly connected
(ii) with only two input symbols
(iii) being a permutation machine
(iv) being an autonomous machine
(v) its outputs being already linearly coded
(vi) its flow table being completely specified

However, the existency of a linear machine to realize a given machine, if

it has already been found, does not guarantee a minimum dimension of realization. It needs further study to obtain the following techniques:

(i) A systematic way to obtain the most economical linear realization for M from the one for M^{τ}.

(ii) When state-splitting is allowed, an efficient method to find a one-to-many linear realization of the minimum possible dimension.

Appendix

Example 1.

Determine whether there exists a linear realization over GF(2) for the machine M shown below:

M	u_o	u_1
f	a	a
e	b	b
d	f	f
c	e	e
b	d	c
a	c	d

$M^{\tau c}$	u_o	u_1	u_2	u_3	
f	a	a	a	b	$\}R_1$
e	b	b	b	a	
d	f	f	e	f	$\}R_2$
c	e	e	f	e	
b	d	c	d	d	$\}R_3$
a	c	d	c	c	

Note that $M^{\tau}=M$

$\gamma = \{\overline{a,b}, \overline{c,d}, \overline{e,f}\}$; $\rho = \{\overline{f,e}, \overline{d,c}, \overline{b,a}\}$

$R_1^o = \begin{bmatrix} a & b \\ b & a \end{bmatrix}$; $R_2^o = \begin{bmatrix} f & e \\ e & f \end{bmatrix}$; $R_3^o = \begin{bmatrix} d & c \\ c & d \end{bmatrix}$

M is one-to-many linearly realizable, because each R_i^o is a standard array. We need state-splitting of Case (i);

$$v = 3$$

$$m = \lceil \log_p v \rceil = \lceil \log_2 3 \rceil = 2$$

$$p^m = 2^2 = 4$$

Splitting:

$$a \longrightarrow a_1, a_2, a_3, a_4$$

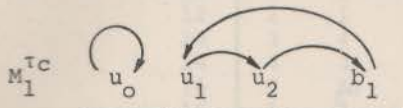
$$b \longrightarrow b_1, b_2, b_3, b_4$$

$$\vdots \qquad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$f \longrightarrow f_1, f_2, f_3, f_4$$

After one operation of Case (i) splitting,

M_1^{Tc} is obtained:



M_1^{Tc}	u_o	u_1	u_2		b_1	
f_1	a_1	a_2	a_3	.	b_1	
f_2	a_2	a_1	a_4	.	b_2	
f_3	a_3	a_4	a_1	.	b_3	
f_4	a_4	a_3	a_2	.	b_4	
e_1	b_1	b_2	b_3	.	a_1	R_1'
e_2	b_2	b_1	b_4	.	a_2	
e_3	b_3	b_4	b_1	.	a_3	
e_4	b_4	b_3	b_2	.	a_4	
d_1	f_1	f_3	e_1	.	f_2	
d_2	f_2	f_4	e_2	.	f_1	
d_3	f_3	f_1	e_3	.	f_4	
d_4	f_4	f_2	e_4	.	f_3	
c_1	e_1	e_3	f_1	.	e_2	R_2'
c_2	e_2	e_4	f_2	.	e_1	
c_3	e_3	e_1	f_3	.	e_4	
c_4	e_4	e_2	f_4	.	e_3	
b_1	d_1	c_1	d_2	.	d_3	
b_2	d_2	c_2	d_1	.	d_4	
b_3	d_3	c_3	d_4	.	d_1	
b_4	d_4	c_4	d_3	.	d_2	
a_1	c_1	d_1	c_2	.	c_3	R_3'
a_2	c_2	d_2	c_1	.	c_4	
a_3	c_3	d_3	c_4	.	c_1	
a_4	c_4	d_4	c_3	.	c_2	

Each R_i' can be formed as a standard array:

M_1^{Tc}		u_o	u_2	u_3		u_3	
f_2	f_1	a_1	a_2	a_3	.	b_1	...
f_1	f_2	a_2	a_1	a_4	.	b_2	
f_4	f_3	a_3	a_4	a_1	.	b_3	
f_3	f_4	a_4	a_3	a_2	.	b_4	
e_2	e_1	b_1	b_2	b_3	.	a_1	
e_1	e_2	b_2	b_1	b_4	.	a_2	
e_4	e_3	b_3	b_4	b_1	.	a_3	
e_3	e_4	b_4	b_3	b_2	.	a_4	
	d_1	f_1	f_3	e_1	.	f_2	...
	d_3	f_3	f_1	e_3	.	f_4	
	c_1	e_1	e_3	f_1	.	e_2	
	c_3	e_3	e_1	f_3	.	e_4	
	d_2	f_2	f_4	e_2	.	f_1	
	d_4	f_4	f_2	e_4	.	f_3	
	c_2	e_2	e_4	f_2	.	e_1	
	c_4	e_4	e_2	f_4	.	e_3	
	b_1	d_1	c_1	d_2	.	d_3	...
	a_1	c_1	d_1	c_2		c_3	
	b_2	d_2	c_2	d_1	.	d_4	
	a_2	c_2	d_2	c_1	.	c_4	
	b_3	d_3	c_3	d_4	.	d_1	
	a_3	c_3	d_3	c_4	.	c_1	
	b_4	d_4	c_4	d_3	.	d_2	
	a_4	c_4	d_4	c_3	.	c_2	

Example 2.

Determine whether there exists a linear realization over GF(2) for the machine M shown below:

M	u_o	u_1	u_2	u_3
11	1	2	3	4
12	2	1	4	3
13	3	4	1	2
14	4	3	2	1
15	5	6	7	8
16	6	5	8	7
17	7	8	5	6
18	8	7	6	5
1	11	15	16	17
2	12	16	15	18
3	13	17	18	15
4	14	18	17	16
5	15	11	12	13
6	16	12	11	14
7	17	13	14	11
8	18	14	13	12

Note that $M^T=M$

M^{TC}	u_o	u_1	u_2	u_3	u_1'	u_2'	u_3'	
11	1	2	3	4	5	6	7	
12	2	1	4	3	6	5	8	
13	3	4	1	2	7	8	5	R_1
14	4	3	2	1	8	7	6	
15	5	6	7	8	1	2	3	$=R^o_1$
16	6	5	8	7	2	1	4	
17	7	8	5	6	3	4	1	
18	8	7	6	5	4	3	2	
1	11	15	16	17	12	13	14	
2	12	16	15	18	11	14	13	
3	13	17	18	15	14	11	12	R_2
4	14	18	17	16	13	12	11	
5	15	11	12	13	16	17	18	$=R^o_2$
6	16	12	11	14	15	18	17	
7	17	13	14	11	18	15	16	
8	18	14	13	12	17	16	15	

$$\gamma=\{\overline{1,2,3,4,5,6,7,8}\ ;\ \overline{11,12,13,14,15,16,17,18}\}$$

The flow table of M^{TC} can not be ordered as a standard array; while each of R_1^o and R_2^o can, hence M is one-to-many linearly realizable.

M^{Tc}	u_o	u_1	u_2	u_2'	u_3'	u_3	u_1'	
11	1	2	3	6	7	4	5	R_1
12	2	1	4	5	8	3	6	
13	3	4	1	8	5	2	7	
14	4	3	2	7	6	1	8	
16	6	5	8	1	4	7	2	
15	5	6	7	2	3	8	1	
18	8	7	6	3	2	5	4	
17	7	8	5	4	1	6	3	
1	11	15	16	13	14	17	12	R_2
5	15	11	12	17	18	13	16	
6	16	12	11	18	17	14	15	
2	12	16	15	14	13	18	11	
3	13	17	18	11	12	15	14	
7	17	13	14	15	16	11	18	
8	18	14	13	16	15	12	17	
4	14	18	17	12	11	16	13	

conflicting columns

We need state-splitting of Case (ii):

u_o	u_1	u_2	u_2'	u_3'	u_3	u_1'		
1	2	3	6	7	4'	5'	R_1	R_1'
2	1	4	5	8	3'	6'		
3	4	1	8	5	2'	7'		
4	3	2	7	6	1'	8'		
6	5	8	1	4	7'	2'		
5	6	7	2	3	8'	1'		
8	7	6	3	2	5'	4'		
7	8	5	4	1	6'	3'		
1'	2'	3'	6'	7'	4	5	R_1	
2'	1'	4'	5'	8'	3	6		
3'	4'	1'	8'	5'	2	7		
4'	3'	2'	7'	6'	1	8		
6'	5'	8'	1'	4'	7	2		
5'	6'	7'	2'	3'	8	1		
8'	7'	6'	3'	2'	5	4		
7'	8'	5'	4'	1'	6	3		
11	15	16	13	14	17'	12'	R_2	R_2'
15	11	12	17	18	13'	16'		
16	12	11	18	17	14'	15'		
12	16	15	14	13	18'	11'		
13	17	18	11	12	15'	14'		
17	13	14	15	16	11'	18'		
18	14	13	16	15	12'	17'		
14	18	17	12	11	16'	13'		
11'	15'	16'	13'	14'	17	12	R_2	
15'	11'	12'	17'	18'	13	16		
16'	12'	11'	18'	17'	14	15		
12'	16'	15'	14'	13'	18	11		
13'	17'	18'	11'	12'	15	14		
17'	13'	14'	15'	16'	11	18		
18'	14'	13'	16'	15'	12	17		
14'	18'	17'	12'	11'	16	13		

The rows in the lower half of R_2' are interchanged to have a standard array:

M_1^{TC}		u_0	u_1	u_2	u_2'	u_3'	u_3	u_1'
11'	11	1	2	3	6	7	4'	5'
12	12'	2	1	4	5	8	3'	6'
13'	13	3	4	1	8	5	2'	7'
14	14'	4	3	2	7	6	1'	8'
16'	16	6	5	8	1	4	7'	2'
15	15'	5	6	7	2	3	8'	1'
18'	18	8	7	6	3	2	5'	4'
17	17'	7	8	5	4	1	6'	3'
11	11'	1'	2'	3'	6'	7'	4	5
12'	12	2'	1'	4'	5'	8'	3	6
13	13'	3'	4'	1'	8'	5'	2	7
14'	14	4'	3'	2'	7'	6'	1	8
16	16'	6'	5'	8'	1'	4'	7	2
15'	15	5'	6'	7'	2'	3'	8	1
18	18'	8'	7'	6'	3'	2'	5	4
17'	17	7'	8'	5'	4'	1'	6	3
1	1	11	15	16	13	14	17'	12'
5'	5'	15	11	12	17	18	13'	16'
6	6	16	12	11	18	17	14'	15'
2'	2'	12	16	15	14	13	18'	11'
3	3	13	17	18	11	12	15'	14'
7'	7'	17	13	14	15	16	11'	18'
8	8	18	14	13	16	15	12'	17'
4'	4'	14	18	17	12	11	16'	13'
8'	8'	18'	14'	13'	16'	15'	12	17
4	4	14'	18'	17'	12'	11'	16	13
3'	3'	13'	17'	18'	11'	12'	15	14
7	7	17'	13'	14'	15'	16'	11	18
6'	6'	16'	12'	11'	18'	17'	14	15
2	2	12'	16'	15'	14'	13'	18	11
1'	1'	11'	15'	16'	13'	14'	17	12
5	5	15'	11'	12'	17'	18'	13	16

References

1. Brzozowski, J. A. and W. A. Davis, "On the Linearity of Autonomous Sequential Machines", IEEE Trans. EC-13, 673-679 (1964).
2. Cohn, M. "A Theorem on Linear Automata", IEEE Trans., EC-13, 52 (1964).
3. Cohn, M. and S. Even, "Identification and Minimization of Linear Machines", IEEE Trans. EC-14, 367-376 (1965).
4. Davis, W. A. "Linear Realization for Permutation Machines". IFIP Congress 68, Edinburgh, August, 1968.
5. Davis, W. A. "The Linearity of Sequential Machines: A Critical Review", IEEE Conference Record 1968, Symposium on Switching and Automata Theory, 427-430.

6. Davis, W. A. and J. A. Brzozowski. "On the Linearity of Sequential Machines". IEEE Trans., EC-15, 21-29 (1966).
7. Elspas, B. "The Theory of Autonomous Linear Sequential Networks". IRE Trans., CT-6, 45-60 (1959).
8. Gill, A. "Analysis and Synthesis of Stable Linear Sequential Circuits". Jour. A.C.M. 12, 141-149 (1965).
9. Gill, A. "Linear Sequential Circuits", N.Y. McGraw-Hill, 1966.
10. Gill, A. "Graphs of Affire Transformation, with Applications to Sequential Circuits", 7th Annual Symposium, Switching and Automata Theory, Berkeley, California, 127-135 (1966).
11. Gill, A. "Synthesis of Linear Sequential Circuits from Input-output Relations", SIAM J. Appl. Math. 16, 216-227 (1968).
12. Gill, A. and C. J. Tan. "The Factorization of Linear Cycle Sets", IEEE Trans. CT-12, 630-632 (1965).
13. Hartmanis, J. "Two Tests for the Linearity of Sequential Machines", IEEE Trans. EC-14, 781-786 (1965).
14. Hartmanis, J. and W. A. Davis, "Homomorphic Images of Linear Sequential Machines", Journal of Computer and System Sciences, 155-165 (1967).
15. Hartmanis, J. and R. E. Stearns, "Algebraic Structure Theory of Sequential Machines", N. J., Prentice-Hall,1966.
16. Marino, P. J., "Indentification and Synthesis of Linear Sequential Machines", B.S.T.J. 47, 343-384 (1968).
17. Srinivasan, C. V. "State Diagram of Linear Sequential Machines", Jour. Franklin Inst., 273, 383-418 (1962).
18. Wang, K. C. "Transition Graphs of Affire Transformations on Vector Spaces over Finite Fields", Jour. of The Franklin Inst., 283, 55-72 (1967).
19. Yau, S. S. and K. C. Wang, "Linearity of Sequential Machines", IEEE Trans., EC-15, 337-354 (1966).
20. Matluck, M. M. and A. Gill, "Decomposition of Linear Sequential Circuits over Residue Class Rings", Jour. of The Franklin Inst., 294, No. 3, 167-180 (1972).