ERRATA

Page 44 - insert just before Theorem 5.2.

"It can be shown that these requirements are sufficient for a valid assignment if and only if \( q \equiv 2^n \), \( 2^n = 1 \).

We therefore make the following definition.

Definition 5.7. \( m \) qbp's \( P_1, P_2, \ldots, P_m \), where \( P_i = \{ b_i; \bar{b}_i \} \) over the set \( S \) containing \( q \) states, are said to be mutually consistent if and only if

\[
\#(b_1 \bar{b}_2 \ldots \bar{b}_m) \leq 2^n - m, \tag{5.13}
\]

where \( m \leq n = \left\lfloor \log_2 q \right\rfloor \) and \( b_j \) represents either block of \( P_j \).

In view of this new definition of consistence, we now consider the former definition to be 'pair-wise consistence' and note that pair-wise consistence is a necessary but not sufficient condition for mutual consistence."
ON THE LINEARITY OF SEQUENTIAL MACHINES

by

Wayne A. Davis, B.S.E.

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Department of Electrical Engineering,
Faculty of Pure and Applied Science,
The University of Ottawa,
Ottawa, CANADA.
September, 1963.
ABSTRACT

The purpose of this thesis is to present a method for determining from the flow table of a sequential machine whether the machine is linear or not. Linear sequential machines are introduced and their mathematical properties are discussed. The behaviour of autonomous circuits is analyzed. Methods for finding the canonical form of a linear machine are described. The input-output behaviour can be described by a transfer function, and a method of finding the transfer function using signal flow graph techniques is developed. The method for determining the linearity of a sequential machine is based on a class of binary partitions of states and inputs. The partitions are used to find a minimal assignment for the secondary variables, and if necessary the input variables, such that the next state and output functions are linear. The binary partitions are examined in detail and the properties necessary to produce a valid assignment are developed.
ACKNOWLEDGMENTS

The author wishes to express his gratitude to Professor Janusz A. Brzozowski for introducing the author to linear sequential machines and providing many stimulating comments and suggestions, without which this thesis could not have been written.

The author also wishes to acknowledge the financial support of the Defence Research Board.
Table of Contents

Abstract ........................................ iii
Acknowledgments ................................. iv
1. Introduction ................................. 1
2. Fundamentals ................................. 3
   2.1 Characterization of Linear Machines ... 8
   2.2 Autonomous Behaviour .................... 11
   2.3 Canonical Forms ........................... 13
   2.4 Companion Matrix ........................... 14
   2.5 Rational Canonical Form .................. 16
3. The Transfer Function ....................... 19
   3.1 Application of Signal Flow Graphs ....... 23
4. The Assignment Problem for Linear Sequential
   Machines .................................. 30
5. Properties of Assignments and Partitions .... 37
6. The Partition Approach to the Assignment
   Problem .................................. 46
7. Autonomous Machines ......................... 51
8. Nonsynonomous Machines ...................... 66
   8.1 Nonsynonomous Machines with Nonlinear
       Output .................................. 80
9. Singular Machines ............................. 93
   9.1 The Autonomous Case ....................... 95
   9.2 The Nonsynonomous Case ................... 96
10. Conclusions and Recommendations for Further
    Research .................................. 99
Bibliography ................................... 101
1. Introduction.

Since the publication of Huffman's paper in 1955, linear sequential machines have been widely studied under various names, such as: linear sequential circuits, linear coding networks and modular sequential networks. Their usefulness has been conclusively demonstrated in the design of error-correcting encoders and decoders. Therefore the majority of the material about linear sequential machines has been written with this important application in mind. However, there is an indication that the theory of linear machines, because of its sound mathematical basis may provide an insight to the theory of sequential machines in general. There are, however, a few problems which may impede any significant progress along this line. Linear machines have not been studied previously from the flow table point of view, which is the starting point of synthesis. There are no methods by which it is possible to determine whether a given sequential machine specified by its flow table is linear or not. Furthermore, in most cases the flow table may be realized by either a linear or nonlinear sequential machine. Hence, the linearity of a sequential machine reduces to the problem of finding an assignment for the machine in which the next state and output functions are linear.

This thesis provides, not only a new approach to this problem, but also a solution for a restricted class of machines, which should prove to be extendable to the general case. We restrict our attention to binary realizations of fully specified sequential machines. The search for linear realizations leads to the use of a class of binary partitions of the set of internal states and inputs of the machine. The method of binary partitions will be shown to
provide an answer to the problem of linearity for the class of machines considered, and will lead directly to a linear assignment and to a linear realization, if they exist. Although some search may be necessary, in general, the systematic analysis presented restricts the work to a practical amount. It should be added, that we are mainly concerned with finding minimal realizations, i.e. realizations using the minimum possible number of internal variables.
2. **Fundamentals**

A *sequential machine*\(^1,28,29\) (sequential circuit, finite automaton) is a dynamic system which satisfies the following conditions:

1. The system is finite and fixed. It has:
   a) A fixed, finite number \(r\) of independent binary inputs.
   b) A fixed, finite number \(s\) of binary outputs.
   c) A fixed, finite number \(q\) of distinct internal states.

2. The system is deterministic:
   a) The next (internal) state is completely determined by the present state and the present input.
   b) The next output is completely determined by the present state and the present input.

3. The behaviour of the system in time is discrete and synchronous; i.e. the input, state, and output histories can be completely described as occurring at discrete moments of time, \(t = 0, 1, 2, \ldots\).

The behaviour of a sequential machine can be completely described by a *flow table* (state table) as shown in Fig. 2.1. The present state is denoted by \(S_i\), the present input by \(I_j\) and the next state and the next output by \(S_{ij}\) and \(Z_{ij}\), respectively. It is assumed that all inputs are distinct, i.e. for no two inputs \(I_j\) and \(I_k\) does \(S_{ij} = S_{ik}\) and \(Z_{ij} = Z_{ik}\), for \(i = 1, 2, \ldots, q\). Similarly, all machines are considered to be in reduced form.

For constant input machines, which includes autonomous machines, a simpler form of the flow table, as shown in Fig. 2.2, will be used. Here, if \(S_j\) is the state at time \(t\), \(S'_j\) is the state at
Fig. 2.1. Flow table of a sequential machine.

time $t + 1$. The state $S^j_{t+1}$ is called the successor (state) of $S_j$, and $S_j$ is a predecessor of $S^j_{t+1}$. In general, every state has a unique successor, but it may have more than one predecessor.

Fig. 2.2. Flow table of an autonomous machine.

A sequential machine is said to be fully specified, if both the next state and output are specified for each present state and input configuration. The discussion will be limited to fully specified machines.
By an assignment for a flow table representing a sequential machine, we mean the assignment of internal or secondary variables to represent the states and the assignment of input or primary variables to represent the inputs of the machine. More will be said about assignments later. The sequential machines will be constructed from combinational gates and unit delay elements. Thus the secondary variables will correspond to the outputs of the delay elements.

It should be noted that throughout this thesis we assume that each machine can be started in any state and we are not concerned with how it gets there.

A linear sequential machine is a finite number of exclusive 'or' gates, unit delays and inputs connected together according to a set of connection rules, to be described later, so as to produce a finite number of outputs. The output of a linear sequential machine can be expressed as a linear function of the present input, a finite number of past inputs and a finite number of past outputs. Although the theory is applicable to a multivalued p-ary logic, where p can be any prime, only the binary case will be considered. In the algebra of linear circuits only two operations are defined: multiplication (\cdot) and addition (+); and their tables are shown:

\[
\begin{array}{ccc}
0 \cdot 0 &=& 0 \\
1 \cdot 1 &=& 1 \\
1 \cdot 0 &=& 0 \\
0 \cdot 1 &=& 0
\end{array}
\]
\[
\begin{array}{ccc}
0 + 0 &=& 0 \\
1 + 1 &=& 0 \\
1 + 0 &=& 1 \\
0 + 1 &=& 1
\end{array}
\]

In the remainder of the thesis the \'.\' for multiplication will be omitted; thus, 'c.b' will be written as 'ab'.
From the multiplication and addition tables, the following properties can be derived, for the binary variables \(a, b, c\):

\[
\begin{align*}
\text{(2.1a)} & \quad a + b = b + a \quad \text{(Commutative Law for \(+\))} \\
\text{(2.1b)} & \quad ab = ba \quad \text{(Commutative Law for \(\cdot\))} \\
\text{(2.2a)} & \quad a + (b + c) = (a + b) + c \quad \text{(Associative Law for \(+\))} \\
\text{(2.2b)} & \quad a(bc) = (ab)c \quad \text{ (Associative Law for \(\cdot\))} \\
\text{(2.3)} & \quad a(b + c) = ab + ac \quad \text{(Distributive Law)} \\
\text{(2.4a)} & \quad a + 0 = a \quad \text{(Identity Law for \(+\))} \\
\text{(2.4b)} & \quad a1 = a \quad \text{(Identity Law for \(\cdot\))} \\
\text{(2.5)} & \quad a + 1 = \overline{a} \quad \text{(Inverse Law)}
\end{align*}
\]

In the physical realization of a linear sequential machine, the two types of elements used are:

a) **Exclusive 'or' gate**—where the output is equal to the sum of the inputs (Fig. 2.3).

\[\begin{array}{c}
a \\
\downarrow \\
b \\
\hline \\
\rightarrow c = a + b
\end{array}\]

Fig. 2.3. Exclusive 'or' gate.

b) **Unit delay**—where the output is equal to the input, but is delayed by one unit of time (Fig. 2.4).

\[\begin{array}{c}
a(t_1) \\
\hline \\
\rightarrow a(t_{1+1}) = a(t_1)
\end{array}\]

Fig. 2.4. Unit delay.
The elements are connected together subject to the following rules:

a) the output of any element may be connected to any number of inputs,

b) no two outputs may be connected together, and

c) every feedback loop contains at least one unit delay.

It is further assumed that any finite number of exclusive 'or' gates may be cascaded without accumulating a time delay, that is, the input to the first of \( k \) cascaded exclusive 'or' gates at time \( t_1 \) produces an output at gate \( k \) at the same time \( t_1 \). The networks will be considered to be synchronous, which means that the contents of the unit delays must all change at the same instant of time. A general linear sequential machine with \( n \)-unit delay, \( r \)-inputs and \( s \)-outputs is of the form shown in Fig. 2.5. Note, however, that the delays, as shown in the figure, may or may not be in feedback loops, that is, an output of a delay may serve directly as an output of the machine.

![Diagram](image)

Fig. 2.5. The general linear sequential machine.
For example, the circuit in Fig. 2.6 is a linear sequential machine having a single binary input \(x\), two unit delays \((y_1, y_2)\) and a single binary output \(z\).

\[
\begin{array}{c}
\text{x} \\
\downarrow \\
+ \quad y_1 \\
\downarrow \\
+ \quad y_2 \\
\downarrow \\
z
\end{array}
\]

**Fig. 2.6.** Machine M2.1.

### 2.1 Characterization of Linear Machines

The (internal) state of a sequential circuit at any time \(t_i\) is defined to be the state of the outputs of the delay elements at time \(t_i\). Therefore, the state (for \(n\) delays, there are \(2^n\) states) of a linear sequential circuit at any time \(t_i\), \(Y(t_i)\), can be expressed as a linear function of the state and the input at time \(t_{i-1}\),

\[
Y(t_i) = F \left[ Y(t_{i-1}), X(t_{i-1}) \right], \quad (2.6)
\]

where \(Y(t_i)\) is an \(n\)-tuple representing the internal state at time \(t_i\), and \(X(t_i)\) is an \(r\)-tuple representing the input at time \(t_i\). Similarly, the output of the circuit at any time \(t_i\), \(Z(t_i)\), can be expressed as another linear function,

\[
Z(t_i) = G \left[ Y(t_i), X(t_i) \right], \quad (2.7)
\]

where \(Z(t_i)\) is an \(s\)-tuple. Hence, the next state function for the \(j^{th}\) delay element, \(y_j\), must be given by the equation,

\[
y'_j = y_j a_{1j} + y_{2j} a_{2j} + \ldots + y_{nj} a_{nj} + x_{1j} b_{1j} + x_{2j} b_{2j} + \ldots + x_{rj} b_{rj}, \quad (2.8)
\]

where \(y'_j = y_j(t_i), y_j = y_j(t_{i-1}), a_{ij}, b_{ij} \in \{0,1\}\).
In a similar manner the $i^{th}$ output, $z_i$, is given by the equation,

$$z_i = y_1c_{i1} + y_2c_{i2} + \cdots + y_nc_{in} + x_1d_{i1} + x_2d_{i2} + \cdots + x_md_{im}, \quad (2.9)$$

where $c_{ij}, d_{ij} \in \{0,1\}$.

If the state of the circuit is expressed by a row matrix of order $n$, and the input is expressed by a row matrix of order $r$, then the next state function is given by the matrix equation,

$$Y' = YA + XB, \quad (2.10)$$

where $A$ is an $(n \times n)$ matrix, and $B$ is an $(r \times n)$ matrix.

The output of the circuit is also described by a similar matrix equation,

$$Z = YC + XD, \quad (2.11)$$

where $C$ is an $(n \times s)$ matrix, and $D$ is an $(r \times s)$ matrix. Obviously for a 'Moore' type machine, where the output is a function of the state only, the matrix $D$ is identically zero.

The matrices $A$, $B$, $C$ and $D$ completely describe the combinatorial circuits in Fig. 2.5, and therefore completely characterize the linear sequential machine.

As an example, for the linear sequential machine shown in Fig. 2.7 the following relations hold:

$$y_1' = y_4 + x,$$
$$y_2' = y_1 + y_4,$$
$$y_3' = y_2 + y_4 + x,$$
$$y_4' = y_3 + x,$$
$$z = y_4 + x.$$
Fig. 2.7. Machine M2.2.

The $A$, $B$, $C$ and $D$ matrices are therefore given by:

$$
A = \begin{pmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0
\end{pmatrix}, \\
B = \begin{pmatrix}
1 & 0 & 1 & 1
\end{pmatrix}, \\
C = \begin{pmatrix}
0 \\
0 \\
0 \\
1
\end{pmatrix}, \\
D = (1).
$$

It should be noted that the matrix $A$, because it describes the interconnections between the delay elements, plays a key role in the operation of a linear sequential machine. This is shown by the following: Given a linear sequential machine $M$ specified by the matrices $A$ and $B$, and a sequence of input vectors $X_0, X_1, X_2, \ldots, X_{n-1}$, then the state of $M$ at any time $t_i$ (where $i = 1, 2, 3, \ldots, n$) is given by:

$$
Y_1 = X_0A + X_0B, \\
Y_2 = Y_1A + X_1B = Y_0A^2 + X_0BA + X_1B,
$$
\[ Y_j = X_0A^3 + X_0BA^2 + X_1BA + X_2B, \]
\[ \ldots \]
\[ Y_m = X_0A^n + X_0BA^{m-1} + X_1BA^{m-2} + \ldots + X_{m-1}B. \] (2.12)

However, for the given input sequence and an initial state \( Y_0 \), each coefficient of \( A^i \) is completely specified. Therefore the state of the machine at any time \( t \) is a linear function of powers of the matrix \( A \).

2.2 Autonomous Behaviour

An autonomous linear sequential machine is a linear sequential machine whose input is zero. From (2.10) the state of the machine at any time \( t_i \) is given by:

\[ Y_i = Y_0A^i = Y_{i-1}A. \] (2.13)

If each state \( Y_i \) of the machine has a unique predecessor (previous state) \( Y_{i-1} \), then the matrix \( A \) must have a unique inverse \( A^{-1} \), such that \( AA^{-1} = A^{-1}A = I \), where \( I \) is the identity matrix.

From matrix theory, a square matrix \( R \), has a unique inverse \( R^{-1} \), if and only if \( R \) is nonsingular, that is, if its determinant, \( |R| \), is different from zero. Every autonomous linear sequential machine in which each state has both a unique predecessor and a unique successor (next state) must consist entirely of cycles. Obviously, if \( Y_0 = 0 \) (the zero state) then \( Y_i = 0 \), for all values of \( i \) and for all matrices \( A \). Hence, every autonomous linear sequential machine contains the trivial zero cycle.

From (2.13) it is seen that if \( Y_i = Y_0 \), then \( A^i = I \), where \( I \) is the identity matrix. Therefore, in order to find the length of the cycles of a linear sequential machine from its characteristic
matrix $A$, of order $n$, without having to find $Y_{i+1}$ for each value of $Y_i$, it is sufficient to find the integral values of $x < 2^n$ such that $A^x = I$.

A maximal period linear sequential machine with $n$ delay elements is one in which the $(2^n - 1)$ non-zero states form a single cycle. A necessary and sufficient condition that a linear sequential machine $M$ be a maximal period linear sequential machine is that:

a) the characteristic polynomial* of $A$ be irreducible, that is, have no nontrivial factors, and

b) it not be a divisor of $\lambda^k + 1$, for any integer $k < 2^n - 1$. However, irreducibility by itself is sufficient to ensure that all the nontrivial cycles in the state diagram are of equal length. Tables of irreducible polynomials up to degree 19 have been published; however, for higher degrees a search for possible factors must be made.

For example, consider the autonomous linear sequential machine $M_{2,5}$ with characteristic matrix

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

From equation (2.14) it is seen that $\Phi(\lambda) = \lambda^3 + \lambda + 1$, which is irreducible, and it can be shown that $\Phi(\lambda)$ is not a divisor of $\lambda^k + 1$ for any integer $k < 7$. Hence, $M_{2,5}$ is a maximal period linear sequential machine with a major cycle of 7 states. This is verified by the state graph of $M_{2,5}$ shown in Fig. 2.8.

Because of the ability of maximal period machines to generate maximum length binary sequences, autonomous linear sequential machines have proved to be quite useful in coding theory for the generation

---

*The characteristic polynomial $\Phi(\lambda)$ of a square matrix $R$ is defined by the equation: $\Phi(\lambda) = |R - \lambda I|$. (2.14)
of cyclic codes.

Fig. 2.3. The state graph of M2.3.

2.3 Canonical Forms

It is possible to have two linear sequential machines which have different internal structures, and yet produce the same output sequence for a given input sequence and initial state. Clearly, the machines have the same behaviour, are indistinguishable, and therefore can be called isomorphic. In isomorphic machines there exists a one-to-one correspondence between states.

Let \( M_a \) and \( M_b \) be two isomorphic machines. Let \( M_a \) and \( M_b \) be both linear, and there exist a one-to-one linear transformation between the states \( Y_a \) of \( M_a \) and the states \( Y_b \) of \( M_b \) such that,

\[
Y_b = Y_a F.
\]

(2.15)

We see that the states of \( M_a \) are related by the equation,

\[
Y'_a = Y_a A_a + X B_a.
\]

(2.16)

Combining (2.15) and (2.16) we get

\[
y_a' F^{-1} = y_b F^{-1} A_a + X B_a.
\]

(2.17)

From this it is seen that,

\[
y'_b = y_b F^{-1} A_a F + X B_a F.
\]

(2.18)
However, the states of $M_b$ are also related by the equation,

$$Y_b = X_b A_b + X_b B_b .$$

Comparing (2.12) with (2.19) it is seen that,

$$A_b = F^{-1} A_a F .$$  

$$B_b = B_a F .$$  

By a similar procedure it can be shown that,

$$C_b = F^{-1} C_a ,$$  

$$D_b = D_a .$$

The matrices $A_a$ and $A_b$ related as in (2.20) by the nonsingular matrix $F$ are called similar matrices$^{30}$.  

Obviously, if for every group of isomorphic machines, there existed a canonical form which had a minimal structure, the analysis of linear sequential machines would be greatly simplified. It has been shown that, indeed, this is the case$^{31}$, and every linear sequential machine can be classified in one of two forms,

a) companion matrix, and

b) rational canonical form.

2.4 Companion Matrix

Every square matrix $R$ satisfies a unique polynomial of lowest degree$^{30}$ of the form,

$$m(R) = R^n + a_{n-1} R^{n-1} + \ldots + a_1 R + a_0 = 0 ,$$

(2.24)

called the minimum polynomial of $R$. An $(r \times r)$ matrix is said to be nongenerative, if the degree of its minimum polynomial is $r$; otherwise the matrix is generative. It has been shown$^{31}$ that every linear sequential machine whose characteristic matrix is nongenerative has a companion matrix which is similar to the characteristic matrix of
the machine. The companion matrix of a minimum polynomial of the form of (2.24) is given\(^3\) by:

\[
C_m = \\
\begin{pmatrix}
0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
& \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & a_1 & a_2 & a_3 & \cdots & a_{n-1}
\end{pmatrix}
\] (2.25)

This is to say that, for every linear sequential machine \(M\) whose characteristic matrix \(A\) is nonderogatory, there exists a matrix \(F\) such that \(C_m = F^{-1}AF\), and the linear sequential machine \(M_0\) having \(C_m\) as its characteristic matrix is isomorphic to \(M\). From (2.25) it is seen that the next state function of each delay element \(y_i\) is a function of only the state of the previous delay element \(y_{i-1}\), the input \(x\), and the state of the last delay element \(y_n\). This is often termed the 'shift register' representation of the machine.

For example consider the linear sequential machine M2.4 shown in Fig. 2.9.

![Diagram](image)

**Fig. 2.9. Machine M2.4.**

M2.4 is described by the following matrices,

\[
A_1 = \\
\begin{pmatrix}
0 & 0 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1
\end{pmatrix},
\]

\[
B_1 = (1 \ 0 \ 1),
\]

\[
C_1 = \begin{pmatrix}
1 \\
1 \\
1
\end{pmatrix},
\]
\[ D_1 = (1) \].

The characteristic polynomial of \( A_1 \) is \( \Phi(\lambda) = \lambda^3 + \lambda + 1 \).

It can be shown that \( \Phi(\lambda) = \omega(\lambda) \).

Therefore a linear sequential machine M2.5 exists which is isomorphic to M2.4, having a characteristic matrix

\[
A_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.
\]

It can be shown from (2.20) that \( A_1 \) and \( A_2 \) are similar matrices related by the nonsingular matrix,

\[
F = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{pmatrix}.
\]

Therefore, from (2.21), (2.22) and (2.23),

\[
b_2 = (0 1 0),
\]

\[
c_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},
\]

\[ D_2 = (1). \]

M2.5 is shown in Fig. 2.10.

---

2.5 Rational Canonical Form

Every linear sequential machine \( \mathcal{M} \) whose characteristic matrix \( A \) is derogatory is isomorphic to a machine having a character-
istic matrix $R$, in what is called the rational canonical form\textsuperscript{31}.

$$R = \begin{pmatrix}
R_1 & 0 & 0 & \ldots & 0 \\
0 & R_2 & 0 & \ldots & 0 \\
0 & 0 & R_3 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & R_n
\end{pmatrix} \quad (2.26)$$

Here, $R_n$ is the companion matrix of the minimum polynomial of $A$, and each other $R_i$ is the companion matrix of a polynomial $p_i(\lambda)$, where each $p_i(\lambda)$ divides $p_{i+1}(\lambda)$. Clearly, $M$ is composed of $m$ smaller machines which are connected only by common inputs and whose outputs are combined to form the output of the total machine $M$.

As an example, consider the linear sequential machine whose characteristic matrix is,

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

The characteristic polynomial of $A$ is,

$$\Phi(\lambda) = \lambda^5 + \lambda^4 + \lambda + 1 = (\lambda + 1)^5.$$  

By considering the powers of $\lambda$ it is seen that, $A^4 = I$, and that,

$$A^3 + A^2 + A + I = 0.$$  

Therefore the minimum polynomial of $A$ is,

$$m(\lambda) = \lambda^3 + \lambda^2 + \lambda + 1.$$  

$A$ is therefore derogatory and has a rational canonical form\textsuperscript{31}:

$$R_A = \begin{pmatrix}
A_1 & 0 \\
0 & A_2
\end{pmatrix},$$  

because from (2.26) we have,
\[ R_n = A_2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \text{ and} \]

\[ A_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \]

In summary, each linear sequential machine \( N \) is isomorphic
in the nondeterogatory case, to a machine whose characteristic matrix
is the companion matrix of \( A \), or in the derogatory case, to a collection
of such machines whose overall characteristic matrix is in the rational
canonical form.
3. The Transfer Function

If the binary input or output sequence of a linear sequential machine is given by \( x_0, x_1, x_2, \ldots, x_n, \ldots \), where \( x_1 \) is the input or output at time \( t_1 \), then the sequence can be expressed as a polynomial in \( D \),

\[
X(D) = x_0 + x_1D + x_2D^2 + \ldots + x_nD^n + \ldots \quad (3.1)
\]

\( D \) is considered to be a delay operator, and, in general multiplication by \( D \) will delay a sequence by one period of time and multiplication by \( D^{-1} \) will advance a sequence by one period of time.

It has been shown\(^{15, 18, 19}\) that the behaviour of a linear sequential machine whose output can be expressed as a function of the present and a finite number of previous inputs, can be described by a polynomial \( P(D) \) in the delay operator, where,

\[
P(D) = a_0 + a_1D + a_2D^2 + \ldots + a_mD^m \quad (3.2)
\]

In other words, for a linear sequential machine with one output and one input, the output sequence, \( Z(D) \), is given in terms of the input sequence, \( X(D) \), by the equation,

\[
Z(D) = X(D)P(D), \quad (3.3)
\]

and the machine is assumed to be in the zero state at time \( t_0 \).

Thus the machine simply multiplies the input sequence by \( P(D) \), and \( P(D) \) is called the transfer function of the machine.

As an example, for the linear sequential machine \( M3.1 \) shown in Fig. 3.1, \( z_m = x_m + x_{m-1} + x_{m-2} + x_{m-5} \), and \( P(D) = 1 + D + D^3 + D^5 \).
Fig. 3.1. Machine M3.1.

However, if the present output of the linear sequential machine is expressed as a function of the present input and a finite number of previous outputs, the output and input sequences are related by an equation in the form

$$Z(D)Q(D) = X(D), \quad \text{(3.4)}$$

$$Z(D) = \frac{X(D)}{Q(D)}. \quad \text{(3.5)}$$

The transfer function of the machine is $\frac{1}{Q(D)}$. For a given input sequence the output sequence is obtained by dividing the input sequence $X(D)$ by the transfer function $Q(D)$ by standard long division of polynomials. If $Q(D)$ is not a factor of $X(D)$ the resulting output sequence will be an infinite sequence with some period $k$.

For example, the linear sequential machine M3.2 shown in Fig. 3.2, has, $z_n = x_n + z_{n-2} + z_{n-4}$, and

$$Q(D) = 1 + D^2 + D^4.$$
Clearly a combination of the two conditions can exist, when the output of the linear sequential machine is a function of the present input and a finite number of previous inputs plus a finite number of previous outputs. In this case the output sequence is related to the input sequence by an equation in the form,

\[ Z(D)Q(D) = X(D)P(D), \text{ or} \]

\[ Z(D) = X(D) \frac{P(D)}{Q(D)}. \]  

Equations (3.6) and (3.7) show the transfer function is \( \frac{P(D)}{Q(D)} \).

As an example, for the linear sequential machine \( M_3 \) shown in Fig. 3.3, \( z_n = x_n + x_{n-1} + x_{n-4} + z_{n-2} + z_{n-3} \), and

\[ \frac{P(D)}{Q(D)} = \frac{1 + D + D^4}{1 + D^2 + D^5}. \]

![Diagram](image)

**Fig. 3.3.** Machine \( M_3 \).

If two linear sequential machines \( H_a \) and \( H_b \) each having one input and one output with transfer functions \( H_a \) and \( H_b \) respectively are cascaded (connected in series), as shown in Fig. 3.4, the output is given by,

\[ Z(D) = X(D)H_a H_b. \]  

Equation (3.8) holds because, \( Z(D) = H_b X_b(D) \), where \( X_b(D) \) is the input to \( H_b \). However, \( X_b(D) = Z_a(D) = X(D)H_a \).

Hence, \( Z(D) = H_b \left[ X(D)H_a \right] = X(D)H_a H_b. \)
Fig. 3.4. Cascade machines.

If $M_a$ and $M_b$ are connected in parallel, as shown in Fig. 3.5, the output is given by,

$$Z(D) = X(D) [ H_a + H_b ] . \quad (3.9)$$

Fig. 3.5. Parallel machines.

This is true because, $Z(D) = Z_a(D) + Z_b(D)$, where $Z_a(D)$ and $Z_b(D)$ are the outputs of $M_a$ and $M_b$ respectively. But $Z_a(D) = X(D)H_a$, and $Z_b(D) = X(D)H_b$. Therefore $Z(D) = X(D)H_a + X(D)H_b = X(D) [ H_a + H_b ]$.

Obviously the reverse is true, for if a linear sequential machine has a transfer function $H$ which can be factored so that

$$H = H_a H_b,$$

the machine may be realized as two machines $M_a$ and $M_b$, with transfer functions $H_a$ and $H_b$ respectively, operating in series.

Similarly, if $H = H_a + H_b$, then the machine may be realized by $M_a$ and $M_b$ operating in parallel.

In the general case of a linear sequential machine with $r$ inputs, and $s$ outputs, the output sequence for the $i$th output will be given by the equation,

$$Z_i(D) = H_{i1}X_1(D) + H_{i1}X_2(D) + \ldots + H_{i1}X_r(D). \quad (3.10)$$
If the \( r \) output sequences are expressed as a vector \( Z \), such that

\[
Z = \begin{pmatrix}
Z_1(p) \\
Z_2(p) \\
Z_3(p) \\
\vdots \\
Z_r(p)
\end{pmatrix},
\]

then the output vector is given in terms of the input vector by the equation:

\[
Z = (H_{ij})X,
\]

where \( H_{ij} \) is the transfer function of the \( i \)th output with respect to the \( j \)th input.

The next section presents a method of obtaining the transfer function for any linear sequential machine using the theory of linear signal flow graphs.

3.1 Application of Signal Flow Graphs

A signal flow graph\(^{25}\)\(^{26}\) is a network of directed branches which connect a set of nodes. Associated with each node is a quantity called the node signal, and associated with each branch is a quantity called the branch gain. A **source** node is one having only outgoing branches, while a **sink** node is one having only incoming branches. A **path** is a succession of connected branches all having the same direction. A **forward path** is a path connecting a source to a sink in which no node appears more than once. A **feedback path** (or loop) is a path which forms a closed loop in which no node appears more than once. The gain of a path is the product of the branch gains.
A signal flow graph is termed linear, if the signal at any node can be expressed as a linear function of the signals at the other nodes. The node signal at any node is the algebraic sum of the incoming signals. The node signal at the sink of a graph having two nodes and one branch, is equal to the product of the branch gain and the source node signal. The gain or transmission of a signal flow graph is the signal at the sink per unit signal at the source.

Each of the elements used in linear sequential machines have distinct parallels in linear signal flow graphs. An exclusive 'or' gate, which has an output equal to the modulo 2 sum of the inputs, may be represented, as shown in Fig. 5.6, by a node of the flow graph, where addition is performed modulo 2.

![Exclusive 'or' gate](image)

**Fig. 5.6.** Exclusive 'or' gate, and node representation.

A unit delay element, which has an output equal to the input delayed by one unit of time, may be represented (Fig. 5.7) by a directed branch having a gain of D (the delay operator).

![Unit delay](image)

**Fig. 5.7.** Unit delay, and branch representation.
A connection from the output of any element to the input of any element may be represented (Fig. 3.8) by a directed branch having a gain of 1.

Fig. 3.8. A connection, and branch representation.

The extension of the method to a multivalued p-nary logic, where p is a prime, is accomplished by representing multiplication by a constant by a directed branch having a gain equal to the constant.

The elementary equivalences of flow graph theory can now be applied to linear sequential machines, remembering the following algebraic rules for module 2 arithmetic,

1) \( a + a = 0 \),

2) \( a = -a \).

The equivalences are shown in Figs. 3.9 through 3.12.

Fig. 3.9. Parallel paths.

Fig. 3.10. Cascade (or series) paths.
Once that the structure of the linear sequential machine has been reduced to a linear signal flow graph, then any of the standard techniques for finding the gain of the graph will yield the desired transfer function. The signal at the sink is the output function, while the signal at the source is the input function. The reduction techniques are sufficiently explained in a variety of references\textsuperscript{25,26} and will not be repeated here; however, a number of examples are shown to illustrate the ease with which the method can be used.

Example 5.1

The flow graph for the linear sequential machine H5.1 in Fig. 5.1 is shown in Fig. 5.13.
\[ \frac{Z(D)}{X(D)} = \mathbb{F}(D) = \left[ \frac{(a + b^2) D^2 + 1}{D + 1} \right] D + 1 \]
\[ = 1 + D + b^2 + D^4. \]

**Fig. 3.13. Flow graph of \( H_3.1 \).**

**Example 3.2**

The flow graph for the linear sequential machine \( H_5.2 \) is shown in **Fig. 3.14.**

\[ \frac{Z(D)}{X(D)} = \frac{1}{1 + D^2 + D^4}. \]

**Fig. 3.14. Flow graph of \( H_5.2 \).**

**Example 3.3**

The flow graph for the linear sequential machine in **Fig. 3.3** is shown in **Fig. 3.15.**

\[ \frac{Z(D)}{X(D)} = \frac{1 + D(1 + b^2)}{1 + D^2 + D^4} = \frac{1 + D + D^4}{1 + D^2 + D^4}. \]
Fig. 3.15. Flow graph of M5.2.

In the preceding examples the transfer function may be written down by inspection from the flow graph, and even directly from the circuit diagram. However, this may not be always straightforward as is demonstrated in the following example.

Example 5.4

For the linear sequential machine M5.4 in Fig. 3.16, the flow graph is shown in Fig. 3.17 and the transfer function is

\[ Z(D) = \frac{(D + 1)(D^2 + 1)D + 1}{1 + \frac{D^3}{1 + D^2 + D^4}} \]

\[ = \frac{D^4 + 1}{D^4 + D^3 + D^2 + 1} = \frac{1}{1 + \frac{1}{D + D^3}}. \]

Fig. 3.16. Machine M5.4.
Fig. 3.17. Flow graph of M5.4.

As an example of the use of transfer functions, consider machine M5.4 and its associated transfer function. Given the input sequence 1 0 1 1 0 1 0 0 1 1 which is represented by the delay polynomial \( x(\Delta) = 1 + \Delta^2 + \Delta^3 + \Delta^5 + \Delta^8 + \Delta^9 \). The transfer function of M5.4 is given by \( \frac{1}{Q(\Delta)} = \frac{1}{1 + \Delta + \Delta^3} \).

The output sequence is therefore given by

\[
Z(\Delta) = \frac{x(\Delta)}{Q(\Delta)} = \frac{1 + \Delta^2 + \Delta^3 + \Delta^5 + \Delta^8 + \Delta^9}{1 + \Delta + \Delta^3}.
\]

We now divide \( x(\Delta) \) by \( Q(\Delta) \) to obtain \( Z(\Delta) \).

\[
\begin{align*}
1 + \Delta + \Delta^3 & \mid \begin{array}{c}
1 + \Delta + \Delta^4 + \Delta^7 + \Delta^9 \\
1 + \Delta^2 + \Delta^5 + \Delta^8 + \Delta^9 \\
1 + \Delta + \Delta^3 \\
\Delta + \Delta^2 + \Delta^4 \\
\Delta + \Delta^2 + \Delta^4 \\
\Delta^4 + \Delta^5 + \Delta^7 \\
\Delta^7 + \Delta^8 + \Delta^{10} \\
\Delta^9 + \Delta^{10} + \Delta^{12} \\
\end{array} \\
\end{align*}
\]

\[
Z(\Delta) = 1 + \Delta + \Delta^4 + \Delta^7 + \Delta^9.
\]

The output sequence is therefore 1 1 0 0 1 0 0 1 0 1 for the given input sequence.
4. The Assignment Problem For Linear Sequential Machines

In section 2.2 and the literature\textsuperscript{8,11,31} a linear machine is called autonomous when it has no input ($x = 0$); then such a machine can be described by:

$$Y' = YA, \quad (4.1)$$
$$Z = YO. \quad (4.2)$$

This terminology seems to be unnecessarily restrictive: it appears logical to allow a machine with a constant input ($x = 1$) to be called autonomous. In a physical realization 0's and 1's are usually represented by electrical signals, and the discrimination between a constant input of 0 and a constant input of 1 has no physical basis. In this thesis a sequential machine will be called autonomous if it has constant inputs. Thus the describing equations now become:

$$Y' = YA + B, \quad (4.3)$$
$$Z = YO + D, \quad (4.4)$$

where $B$ is a $(1 \times n)$ matrix (row vector) and $D$ is a $(1 \times s)$ matrix. Of course, this case includes the case with $x = 0$, when $B = D = 0$.

To illustrate this extended class of autonomous linear machines consider the machine $M4.1$ of Fig. 4.1. The state graph of $M4.1$ is shown in Fig. 4.2.

![Fig. 4.1. Machine M4.1 with constant input.](image-url)
It is soon that the machine has a cycle of length 4. Such a behaviour cannot be obtained from a machine with zero input, since every machine with zero input must have the trivial zero cycle (the state (0,0,...,0) returning to itself). The machine of Fig. 4.1 is described by the equations:

\[(y_1', y_2') = (y_1, y_2) \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + (0, 1),\]

\[z = (y_1, y_2) \begin{pmatrix} 1 \\ 1 \end{pmatrix} + (0).\]

Consider now the flow table of Fig. 4.2.

<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
<th>y_1</th>
<th>y_2</th>
<th>y_3</th>
<th>y_1'</th>
<th>y_2'</th>
<th>y_3'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>B</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

a) Flow table.  b) Assignment 1.  c) Assignment 2.

In Fig. 4.2 b), Assignment 1 results in the next state functions:

\[y_1' = y_2 + 1 = \overline{y_2}, \quad y_2' = y_1 + y_3, \quad y_3' = y_2 + y_3,\]

and is linear. (\(\overline{y}\) denotes the complement of \(y\), \(x\overline{y}\) denotes inclusive 'or' and \(xy\) denotes 'and'). On the other hand, Assignment 2 of Fig. 4.2 b) results in the next state functions:
\[ y_1' = 1 + y_2, \]
\[ y_2' = \bar{y}_1(y_2 + y_3) \lor y_1\bar{y}_2 = y_1 + y_2 + y_3 + y_1y_2, \]
\[ y_3' = \bar{y}_1y_2\bar{y}_3 \lor y_1(y_2 \lor y_3) = y_1 + y_2 + y_3, \]
where \( y_2' \) and \( y_3' \) are nonlinear\(^5\).

Clearly one could try to discover whether a flow table can be linearly realized by trying all possible assignments. This may be suitable for tables with a few states; however, the number of different assignments soon becomes very large and other methods must be developed.

If \( n \) is the smallest integer such that \( 2^n \geq q \), then for a flow table of \( q \) states there are \( 27 \)
\[ N = \frac{(2^n - 1)!}{(2^n - q)!n!} \]
different assignments of \( n \) variables.

From the algebraic point of view, if the flow table of the machine contains \( q \) states \( Y = \{ y_1, y_2, \ldots, y_q \} \) each state \( y_i \) having a successor state \( y_i' \in Y \), then we must find matrices \( A \) and \( B \) such that \((4.3)\) is satisfied for all \( y_i \in Y \), and each state is represented uniquely in the assigned variables. If there exists a trivial zero-cyclo, \( y'_1 = y_1 \), then we can simplify the problem by saying that \( B = 0 \).

If we consider only machines having a nonsingular matrix \( A \), then all states appear in cycles of length \( \in \mathbb{C} = \{ c_1, c_2, \ldots, c_k \} \), where \( c_1 + c_2 + \ldots + c_k = q \). We are therefore looking for matrices \( A \) and \( B \) such that,
\[ y_i = y_iA^j + BA^{j-1} + BA^{j-2} + \ldots + BA + B, \quad (4.6) \]
for all \( j \in \mathbb{C} \). Again, if \( B = 0 \), the problem is simplified, so that \((4.6)\) becomes,
\[ y_i = y_iA^j. \quad (4.7) \]
This appears to be a difficult problem; however, we can approach the problem in a different manner. If we consider the canonical form\(^{31}\)
(shift register representation) as a counter in which the state is an
(n-1)th degree polynomial $y$ in an indeterminate $x$, then counting up
on the counter amounts to multiplying $y(x)$ by $x$, adding a constant
input polynomial $c(x)$ and reducing modulo $m(x)$, an $n$th degree monic poly-
nomial, which is the minimum polynomial of the machine. We must
then find $m(x)$ and $c(x)$ such that
$$y'(x) = x y(x) + c(x) \mod m(x), \quad (4.8)$$
for all states $y(x) \in Y$. In terms of cycle lengths the following
equation must be satisfied,
$$y(x) = x^j y(x) + x^{j-1} c(x) + x^{j-2} c(x) + \ldots + xc(x) + c(x) \mod m(x), \quad (4.9)$$
for all $j \leq 0$. Again the problem is simplified if a $y(x)$ exists
such that $y'(x) = y(x)$, then (4.6) and (4.9) reduce to,
$$y'(x) = x y(x) \mod m(x), \quad (4.10)$$
$$y(x) = x^j y(x) \mod m(x). \quad (4.11)$$

It is worth noting that Fitzpatrick has shown that all
autonomous sequential machines consisting entirely of cycles can be
realized linearly. However, these realizations are minimal only in
the sense that they use the smallest number of delay elements for
the machine to be linear without constant inputs. In general, two
distinct cycles are realized by two distinct sets of elements such
that interconnections are made only between elements of the same set.
As an example of this method, consider the machine in Fig. 4.2 given by the
flow table in Fig. 4.4. Using the methods given by Fitzpatrick,
we see that the assignment given in Fig. 4.5 and the minimum polynomial,
$m(x) = x^4 + x^3 + x + 1$, provide a realization using 4 delay elements.
\begin{align*}
\begin{array}{c|c}
S & S' \\
\hline
A & B \\
B & C \\
C & D \\
D & E \\
E & F \\
F & A \\
\end{array}
\end{align*}

Fig. 4.4. Machine \#4.2.

\begin{align*}
\begin{array}{c|c|c}
S & y(x) & Y = (y_1, y_2, y_3, y_4) \\
\hline
A & 1 & 1000 \\
B & x_2 & 0100 \\
C & x_3 & 0010 \\
D & x_3^2 + x_2 + 1 & 0001 \\
E & x_2^3 + 10 & 1101 \\
F & x_2^2 + 1 & 1011 \\
\end{array}
\end{align*}

Fig. 4.5. Assignment 1 for \#4.2.

From (4.5) and using an exhaustive search, we see that if,

\begin{align*}
A &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \\
B &= \begin{pmatrix} 0 & 1 & 0 \end{pmatrix},
\end{align*}

the assignment of Fig. 4.6 satisfies the flow table using only 3 delay elements.

\begin{align*}
\begin{array}{c|c|c}
S & Y = (y_1, y_2, y_3) \\
\hline
A & 000 \\
B & 010 \\
C & 011 \\
D & 111 \\
E & 101 \\
F & 100 \\
\end{array}
\end{align*}

Fig. 4.6. Assignment 2 for \#4.2.

Similarly from (4.11) we note that Assignment 2 is equivalent to Assignment 3 in Fig. 4.7 where \( m(x) = x^5 + 1 \) and \( c(x) = x \).
\begin{center}
\begin{tabular}{l|l}
\textbf{S} & \textbf{y(x)} \\
\hline
A & 0 \\
B & x^2 \\
C & x^2 + x \\
D & x^2 + x + 1 \\
E & x^2 + 1 \\
F & 1 \\
\end{tabular}
\end{center}

Fig. 4.7. Assignment \( \delta \) for \( \Delta 4.2. \)

In the nonautonomous case the complexity of the problem is obviously increased, for, the nonautonomous machines are described by

\[ Y' = YA + XB, \]
\[ Z = YC + XD. \]

Again to retain generality we permit constant inputs; however, there is no need to add a constant input matrix. For an \( r \) input vector \( X \), we merely add a constant as on \( (r + 1) \) th input and the input vector for the general case becomes \( X = (x_1, x_2, \ldots, x_r, 1). \) Similarly, \( B \) becomes an \( (r + 1) \times n \) matrix, and \( D \) becomes an \( (r + 1) \times s \) matrix.

Clearly one could try to find a linear assignment for a nonautonomous machine by considering each column of the flow table as an autonomous machine, finding an assignment for each autonomous case and then relating the inputs. This method will provide a solution; however, the process may be lengthy, for, the flow table may be linear column by column, but not as a unit. Therefore a more direct and unified approach is desirable. Srinivasan\textsuperscript{35,36} presents an algorithm by which one can determine the linearity of a restricted class of sequential machines. This method is not only lengthy, but can be applied to only a limited class of sequential machines. The main classes of machines excluded from the algorithm are:

\begin{itemize}
  \item[a)] autonomous machines,
\end{itemize}
b) singular machines,
c) machines having \( q \neq 2^n \) states, and
d) machines, where the states appear in two disjoint parts, there being no transitions between states in one part to those in the other.

It is unjustified\(^{35,36}\) to say that this class of machines are of little practical interest. An example of one such machine is \( \#5 \) in Fig. 8.4, where the smaller, disconnected part of the machine is omitted from the flow table.

Since the algebraic approach to the problem of a minimal realization does not appear to lead to a solution other than by enumeration, and the Srinivasan approach is lengthy and restricted, we will develop the partition approach presented in the following sections. While this approach does have its disadvantages, it is different and hence should provide a better understanding to the problem in general.
5. Properties of Assignments and Partitions

Consider a flow table having q states, $S_1, S_2, \ldots, S_q$.

If we wish to realize the flow table by a circuit constructed from binary devices, we must represent each internal state $S_i$ by an n-tuple of 0's and 1's; this n-tuple specifies the values of the secondary variables $y_1, y_2, \ldots, y_n$. Any correspondence of states with n-tuples will be called an assignment, as long as an n-tuple is specified for each state. An assignment will be called valid, if no two states have the same n-tuple assigned to them. Since there are $2^n$ distinct n-tuples, it is clear that a valid assignment requires that $2^n \leq q$, or, if $\lceil \log_2 q \rceil$ denotes the smallest integer $\geq \log_2 q$, then $n = \lceil \log_2 q \rceil$ secondary variables are needed. An assignment is called minimal, if it is valid and $n = \lfloor \log_2 q \rfloor$. In this thesis we shall be concerned only with minimal assignments.

Fig. 5.1 illustrates these definitions: assignment a) is not valid, b) is valid, but not minimal and c) is minimal.

<table>
<thead>
<tr>
<th>States</th>
<th>$T_1F_2$</th>
<th>$T_2F_2$</th>
<th>$F_2F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>001</td>
<td>000</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>010</td>
<td>011</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>110</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>010</td>
<td>111</td>
<td>11</td>
</tr>
</tbody>
</table>

Fig. 5.1. Assignments.

It is clear that, in any assignment, each variable $y_1$ divides the set of states into two subsets, the subset $b_1$ of k states for which $y_1 = 1$, and the subset $b_\bar{1}$ of $(q - k)$ states for
\[ x = 1 \]

**Definition:** A function \( f \) is said to be continuous at a point \( c \) if for every \( \varepsilon > 0 \), there exists a \( \delta > 0 \) such that

\[ |x - c| < \delta \Rightarrow |f(x) - f(c)| < \varepsilon \]

for all \( x \) in the domain of \( f \) satisfying \( 0 < |x - c| < \delta \).

The following theorem will be useful for the discussion on

Proof: Let \( c \) be an arbitrary point in the domain of \( f \). We must show that \( f \) is continuous at \( c \). Fix \( \varepsilon > 0 \). Since \( f \) is uniformly continuous on \( [a, b] \), there exists a \( \delta > 0 \) such that

\[ |x - y| < \delta \Rightarrow |f(x) - f(y)| < \frac{\varepsilon}{b-a} \]

for all \( x, y \) in \( [a, b] \). Thus, if \( x \) and \( y \) are two points in \( [a, b] \) such that \( |x - c| < \delta \) and \( |y - c| < \delta \), then

\[ |x - y| < \delta \Rightarrow |f(x) - f(y)| < \frac{\varepsilon}{b-a} \]

and hence

\[ |f(x) - f(c)| < \varepsilon \]

for all \( x \) in \( [a, b] \) satisfying \( |x - c| < \delta \). This completes the proof.

**Remark:** A function \( f \) is uniformly continuous on \( [a, b] \) if and only if it is continuous on \( [a, b] \).
advantage of using binary partitions is their simplicity compared to the internal binary variables. The relation\( B_j \) is able to specify the partition completely within \( B_j \). For example, the qbp \( P_1 \) above is completely specified by the partition\( B_1 = (C,D) \). Therefore from now on we will no longer specify the complete partition, but refer only to the partition \( \bar{B}_1 = \frac{e_{11} + e_{12}}{2} \) determined by the block \( b_1 = (C,D) \).

We shall now examine the properties of partitions defined by the secondary variables assigned to a set of terms.

Lemma 5.1. In any minimal assignment of secondary variables \( y_1, y_2, \ldots, y_n \), each \( y_i \) defines a partition \( P_i \), over the variable \( n \) which is a qbp.

Proof: There are \( 2^n \) distinct \( n \)-tuples of D's and L's, \( n \text{-tuplets} \). \( (2^n - 1) \quad \) having \( y_1 = 0 \), the other half having \( y_1 = 1 \). For a valid assignment for \( q \) states we must have \( 2^{n-1} < q \leq 2^n \). We examine at most \( 2^{n-1} \) \( n \)-tuples with \( y_1 = 0 \); hence the definition of the partition \( P_i \) defined by \( y_i \), satisfies \( \#(P_i) \leq 2^{n-1} \). The number of \( n \)-tuples with \( y_1 = 1 \) is clearly \( q - 2^{n-1} \). Hence, \( q - 2^{n-1} \leq \#(P_i) \leq 2^{n-1} \), for any partition defined by a secondary variable \( y_i \). Therefore \( P_i \) is a qbp.

Definition 5.4. Two qbp's \( P_1 = \left\{ b_{ij} ; \bar{b}_{ij} \right\} \) and \( P_2 = \left\{ b_{ij} ; \bar{b}_{ij} \right\} \) are said to be consistent, if and only if the number of \( e_{12} \)-terms in the intersection of either block of \( P_1 \) with either block of \( P_2 \) does not exceed \( 2^{n-2} \).
In other words the following conditions must be satisfied:

\[ \#(b_1 b_2) \leq 2^{n-2}, \]  \hspace{1cm} (5.2a)
\[ \#(b_1 b_2) \leq 2^{n-2}, \]  \hspace{1cm} (5.2b)
\[ \#(b_2 b_2) \leq 2^{n-2}, \]  \hspace{1cm} (5.2c)
\[ \#(b_1 b_2) \leq 2^{n-2}, \]  \hspace{1cm} (5.2d)

For example, let \( S = \{A,B,C,D,E,F\} \) and let \( b_1 = (A,B,C) \), \( b_2 = (D) \) and \( b_3 = (E,F) \). Then one can easily verify that \( P_1 \) is consistent with \( P_2 \) and \( P_2 \) is consistent with \( P_3 \), but \( P_1 \) is not consistent with \( P_3 \). This example also demonstrates that the consistency relation is not transitive.

**Lemma 5.2.** In any minimal assignment, the qbp's \( P_1, P_2, \ldots, P_n \), defined by the secondary variables \( y_1, y_2, \ldots, y_n \) are mutually consistent.

Proof: To prove that the n qbp's are mutually consistent means to prove that any two are consistent. Let \( P_i \) and \( P_j \) be any two qbp's defined by the variables \( y_i \) and \( y_j \). There are \( 2^{n-2} \) n-tuples having \( y_i = 1 \) and \( y_j = 1 \). Hence the maximum number of states in \( (b_1 b_j) \) is \( 2^{n-2} \). The same argument applies to all other combinations of blocks from \( P_i \) with blocks of \( P_j \). Hence \( P_i \) is consistent with \( P_j \).

It is easy to show that consistency is not a sufficient condition for a valid assignment. For example, let \( S = \{A,B,\ldots,H\} \), and let \( b_1 = (A,B,C,D) \), \( b_2 = (A,B,E,F) \) and \( b_3 = (A,B,G,H) \). One can easily verify that the corresponding qbp's are mutually consistent. If \( y_j = 1 \), for states \( S \in b_j \), then the assignment of Fig. 5.2 results.
<table>
<thead>
<tr>
<th></th>
<th>(y_1)</th>
<th>(y_2)</th>
<th>(y_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fig. 5.2.** An invalid assignment.

It is clear that this assignment is invalid, because the states are not uniquely determined by their corresponding n-tuples.

**Definition 5.3.** The sum of two qbp's, \(P_1 = \{b_1; \overline{b_1}\} \) and \(P_2 = \{b_2; \overline{b_2}\} \), is defined by:

\[
P_1 + P_2 = \{b_1 + b_2; \overline{b_1 + b_2}\}.
\]  
(5.3)

It can be easily verified that the sum is independent of the order of the qbp blocks, i.e., that \(\{b_1; \overline{b_1}\} + \{b_2; \overline{b_2}\} = \{b_2; \overline{b_2}\} + \{b_1; \overline{b_1}\} = \{b_1; \overline{b_1}\} + \{b_2; \overline{b_2}\} = \{b_2; \overline{b_2}\} + \{b_1; \overline{b_1}\} = \{b_1 + b_2; \overline{b_1 + b_2}\} \).

(5.4)

**Lemma 5.3.** The sum of two consistent qbp's \(P_1\) and \(P_2\) is a qbp consistent with both \(P_1\) and \(P_2\).

**Proof:** \(\#(b_1 + b_2) = \#(b_1 \overline{b_2} v \overline{b_1} b_2) = \#(b_1 \overline{b_2}) + \#(\overline{b_1} b_2),\) where ' + ' for numbers indicates ordinary addition. Hence, from (5.2) we have, \(\#(b_1 + b_2) \leq 2^{n-2} + 2^{n-2} = 2^{n-1}.\)
Also \[ \#(b_1 \uplus b_2) = q - \#(b_1 b_2) - \#(b_1 \overline{b_2}). \]

Again from (5.2) we have, \[ \#(b_1 \uplus b_2) = q - 2^{n-2} - 2^{n-2} = q - 2^{n-1}. \]

Hence \( \#(b_1 \uplus b_2) \) satisfies the condition of equation (5.1) and \( P_1 + P_2 \) is a qbp. Now consider \( P_1 \) and \( (P_1 + P_2) \). We have,

\[ \#(b_1 b_2) = \#(b_1 b_1 b_2 \lor b_1 \overline{b_2}) = \#(b_1 \overline{b_2}) \leq 2^{n-2}, \]

since \( P_1 \) and \( P_2 \) are consistent. A similar argument applies for the remaining inequalities, and hence \( P_1 \) is consistent with \( P_1 + P_2 \).

By symmetry we can say that \( P_2 \) is consistent with \( P_1 + P_2 \), which completes the proof of the lemma.

As an example, consider \( S = \{A, B, C, D, E, F\} \), \( P_1 \) and \( P_2 \) given by \( b_1 = (A, B, C) \) and \( b_2 = (A, B, E, F) \) respectively. Then \( P_1 + P_2 \) is given by \( b_1 + b_2 = (C, E, F) \), and is a qbp which is consistent with both \( P_1 \) and \( P_2 \), since \( P_1 \) and \( P_2 \) are consistent.

The following properties of addition of qbp's follow from the definitions:

\[ P_1 + P_2 = P_2 + P_1 \] (Commutative Law), \hspace{1cm} (5.5)

\[ P_1 + (P_2 + P_3) = (P_1 + P_2) + P_3 \] (Associative Law), \hspace{1cm} (5.6)

\[ P_1 + 0 = P_1 \] (Identity Law). \hspace{1cm} (5.7)

where \( 0 \) represents the trivial binary partition \( \{\emptyset, I\} \).

Similarly, the following properties of addition of qbp blocks follow from the definitions:

\[ b_1 + b_2 = b_2 + b_1 \] (Commutative Law), \hspace{1cm} (5.8)

\[ b_1 + (b_2 + b_3) = (b_1 + b_2) + b_3 \] (Associative Law), \hspace{1cm} (5.9)

\[ b_1 + \emptyset = b_1 \] (Identity Law), \hspace{1cm} (5.10)

\[ b_1 + I = \overline{b_1} \] (Inverse Law). \hspace{1cm} (5.11)
Definition 5.6. Let \( P_1, P_2, \ldots, P_n \) be \( n \) mutually consistent qbp's.

Then \( P_1, P_2, \ldots, P_n \) are (linearly) independent if and only if

\[
\sum_{i=1}^{n} a_i P_i = 0,
\]

implies that \( a_i = 0 \), for all \( i \), where \( a_i \in \{0,1\} \) and \( \text{OP}_i = 0 \),

\[
\lambda P_i = P_i.
\]

For example, let \( S = \{A, B, C, \ldots, G\} \), \( P_1, P_2 \) and \( P_3 \) be given by \( b_1 = (A, B, C) \), \( b_2 = (A, C, B, F) \) and \( b_3 = (A, C, D, G) \) respectively. It can be shown that \( P_1, P_2 \) and \( P_3 \) are mutually consistent, but not independent, for \( P_1 + P_2 + P_3 = 0 \). However, it is easily verified that \( P_1, P_2 \) and \( P_4 \) given by \( b_4 = (A, B, D, E) \) are mutually consistent and independent.

The above definitions and results lead to the following theorem.

Theorem 5.1. Given \( n \) mutually consistent qbp's \( P_1, P_2, \ldots, P_n \) which are linearly independent, the set of all linear combinations,

\[
\sum_{i=1}^{n} a_i P_i,
\]

is a vector space of dimension \( n \) over the field \( \mathbb{F}_2 \).

Proof: The properties required for a vector space are easily verified.

Note that elements of this vector space are mutually consistent in view of Lemma 5.3.
Lemma 5.4. In any minimal assignment of \( n \) variables \( y_1, y_2, \ldots, y_n \) for \( q \) states, the \( q \) qbp's \( P_1, P_2, \ldots, P_n \) defined by the \( n \) variables are independent.

Proof: If any state \( S_j \) has been assigned the \( n \)-tuple \( (y_1, y_2, \ldots, y_n) \), then we can let the qbp, \( P_i = b^x_i; b^\bar{x}_i \) defined by the variable \( y_1 \), be given by the rule, \( S_j \in b^x_i \) if \( y_1 = 1 \), \( S_j \in b^\bar{x}_i \) if \( y_1 = 0 \). Suppose the \( q \) qbp's are not independent, then we can say that at least one of them, \( P_i \), can be expressed as a linear combination of the others. Therefore, \( y_1 \), which determines \( P_i \), can also be expressed as a linear combination of the other variables. Hence, each of the \( q \) states can be uniquely determined by an \((n - 1)\)-tuple \((y_1, y_2, \ldots, y_{k-1}, y_k + 1, \ldots, y_n)\); however, \((n - 1)\) variables uniquely determine \(2^{n-1}\) states which violates the condition that \( q > 2^{n-1} \). Hence the \( n \) qbp's \( P_1, P_2, \ldots, P_n \) must be independent.

We can now summarize the properties of partitions in a minimal assignment by saying that the partitions are:

a) qbp's. (Lemma 5.1)

b) mutually consistent. (Lemma 5.2)

c) linearly independent. (Lemma 5.4)

Theorem 5.2. Given \( n = \lceil \log_2 q \rceil \) mutually consistent and independent qbp's, \( P_j = \{ b^x_j; b^\bar{x}_j \} \), \( j = 1, 2, \ldots, n \), there exists a minimal assignment of \( n \) internal variables \( y_j \) based on the qbp's \( P_j \), for \( q \) states, where \( 2^{n-1} < q \leq 2^n \).
Proof: For any state $S_i$, assign $y_j = 1$ if $S_i \in B_j$ and $y_j = 0$ if $S_i \in \overline{B_j}$. Since the $P_j$ are mutually consistent and independent, all $q^n$-tuples of $y_j$ values are distinct and hence the assignment is valid.

The assignment problem can now be stated as follows:

given $q$ states, to find a minimal assignment, find $n = \lfloor \log_2 q \rfloor$

mutually consistent and independent qbp's.

Note that if we have $q$ states, then the number of combinations of
$k$ of them is \( \binom{q}{k} = \frac{q!}{k!(q-k)!} \). Since a qbp is not dependent upon the
order of the blocks, i.e. \( \{b_1; \overline{b_2}\} = \{\overline{b_1}; b_2\} \), the total number of
distinct qbp's for $q$ states is given by

\[
N_q = \frac{1}{2} \sum_{i=L}^{n-1} \binom{q}{i} \tag{5.13}
\]

where $n = \lfloor \log_2 q \rfloor$, $L = q - 2^{n-1}$.

This will now be applied in the following section to
determine the existence of linear assignments.
6. The Partition Approach to the Assignment Problem

Before directly applying the results of the previous section in order to find out whether a machine is linear or not, the flow table of the machine is checked to ensure that it satisfies the first necessary condition of linearity (to be described). If the machine does not comply with the condition then it is established that the machine is not linearly realizable with the minimum number of delay elements. The first necessary condition is given in the following theorem:

Theorem 6.1. For every linear sequential machine $M$ having $n$ delay elements and a characteristic matrix $A$ of rank $\alpha$, where $n \geq \alpha \geq 0$, the flow table containing $q \leq 2^n$ distinct states has:

1. in any single input column:
   a) $d$ distinct next states, where $d \leq 2^{\alpha}$, and
   b) each next state cannot be repeated more than $2^{n-\alpha}$ times,

2. a unique one-to-one correspondence between states of any two input columns of the flow table.

Proof: Consider $M$ as a machine having $2^n$ internal states $(2^n-q) < 2^{n-1}$ of which have been deleted from the flow table. The behaviour of $M$ can be described by the matrix equation,

$$Y' = YA + XB.$$  \hspace{1cm} (6.1)

Hence for any input $X_j$ we have

$$Y' = YA + X_jB.$$ \hspace{1cm} (6.2)

First suppose $B = 0$; then we have:
\[ Y' = YA. \]  

(6.3)

Since $A$ has rank $e$, by a theorem in matrix theory, $A$ is equivalent to a matrix $A_0$ such that

\[ A_0 = PAQ, \]  

(6.4)

where $P$ and $Q$ are nonsingular matrices, and

\[ A_0 = \begin{pmatrix} I_e & 0 \\ 0 & 0 \end{pmatrix}, \]  

(6.5)

where $I_e$ is the identity matrix of rank $e$. From (6.4) we have

\[ A = P^{-1}A_0Q^{-1}. \]  

(6.6)

Combining (6.3) and (6.6) we see that

\[ Y' = YP^{-1}A_0Q^{-1}, \]  

(6.7)

or

\[ Y'Q = YP^{-1}A_0. \]  

(6.8)

Let

\[ Y' = Y'Q, \]  

(6.9)

\[ Y_e = YP^{-1}. \]  

(6.10)

Hence

\[ Y'_e = YPA_0 e. \]  

(6.11)

Now (6.11) can be considered to be the next state equation of an autonomous machine $M_0$, the states of which are related to the states of $M$ by (6.9) and (6.10). Note from (6.11) and (6.5) that the next states $Y'_e$ of $M_0$ are functions of only $e$ of the variables of the present state $Y_e$. Hence, $M_0$ has a maximum of $2^e$ distinct next states; however, $I_e$ is nonsingular, therefore $M_0$ has exactly $2^e$ distinct next states. Since the next state function of $M_0$ is independent of $(n-e)$ of the variables, there are $2^{n-e}$ distinct present states of $M_0$ which have the same next state. Hence each state of $M_0$ which appears in the next state column, is repeated $2^{n-e}$ times.
Since $P$ and $Q$ are nonsingular, the correspondences between the states of $M$ and $M_0$, described by (6.9) and (6.10) are one-to-one. Therefore, for $X_jM$ has $2^g$ distinct next states, each state being repeated $2^{n-g}$ times.

Removing the restriction that $B = 0$, it is seen that the transformation

$$Y^* = Y + X_j B,$$  \hspace{1cm} (6.12)

is one-to-one, and the condition holds for all $X_j B$, and hence any input $X_j$.

Finally, by deleting rows of the flow table, it is obvious that the number $d$ of distinct next states, under $X_j$, must be less than or equal to $2^g$, and the number of repetitions must be less than, or equal to $2^{n-g}$, which completes the proof of the first part of the theorem.

Now consider a distinct second input $X_1$. From (6.12) the next states for $X_1$ are related to $Y$ by the transformation

$$Y^+ = Y + X_1 B,$$  \hspace{1cm} (6.13)

which is again one-to-one. Combining (6.12) and (6.13) we have

$$Y^+ = Y^* + X_j B + X_1 B.$$  \hspace{1cm} (6.14)

Again, the transformation given by (6.14) is one-to-one, and this completes the proof of the theorem.

In summary, a flow table of a sequential machine satisfies the first necessary condition of linearity when, if the table contains $g$ distinct states, and any input column contains $d$ distinct next states, the maximum number of times that a state can be repeated is

$$\frac{2^{n-g}}{d} = \left\lfloor \log_2 d \right\rfloor,$$
and \( e = \lfloor \log_2 d \rfloor \), and for every pair of inputs there exists a one-to-one transformation between the next states.

For example, consider the nonautonomous machine M6.1 given in Fig. 6.1.

\[
\begin{array}{c|ccccc}
   & 1 & 2 & 3 & 4 & 5 \\
\hline
A & B & A & C & I & F \\
B & B & A & C & I & F \\
C & D & C & A & E & B \\
D & E & G & H & D & I \\
E & B & A & C & I & F \\
F & B & A & C & I & F \\
G & D & C & A & E & B \\
H & E & G & H & D & I \\
I & E & G & H & D & I \\
\end{array}
\]

Fig. 6.1. Machine M6.1.

From the flow table we see that \( q = 9 \), \( d = 3 \); therefore \( n = 4 \), \( e = 2 \). The maximum number of times that a state is repeated in any one column is 4, and \( 2^{n-e} = 4 \). It is easily verified that there exists a one-to-one correspondence between next states for any two inputs. Hence, M6.1 satisfies the first necessary condition of linearity. However, for the sequential machine M6.2 in Fig. 6.2, it can be shown that the machine does not satisfy the first necessary condition. In this case, \( q = 6 \), \( d = 3 \), and state 0 is repeated 3 times for input 1, which is greater than \( 2^{n-e} = 2 \); hence Part 1) of the condition is not satisfied.
Also the correspondence between the next states under input 2 and input 3 is not one-to-one; hence Part 2) is not satisfied.

<table>
<thead>
<tr>
<th>S</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
<td>F</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>A</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>A</td>
<td>C</td>
<td>E</td>
</tr>
<tr>
<td>F</td>
<td>C</td>
<td>A</td>
<td>D</td>
</tr>
</tbody>
</table>

Fig. 6.2. Machine M6.2.

If \( e < n \), we say that the machine is singular, in which case \( p < q \). Alternately, if \( e = n \), the machine is said to be nonsingular, and \( p = q \). We will first consider the nonsingular case, then proceed to the singular case.
7. Autonomous Machines

In the nonsingular autonomous case we consider only those flow tables in which each state appears in the next state column once and only once. Clearly, each state has both a unique successor and a unique predecessor.

**Definition 7.1.** Given an autonomous sequential machine and the qbp \( P = \{ b \}; \overline{b} \}, \) where \( b = (S_1, S_2, \ldots, S_n) \), \( \sigma P \) the successor of \( P \) is a qbp defined by \( \sigma P = \{ b; \overline{\sigma b} \} \), where \( \sigma b = (S'_1, S'_2, \ldots, S'_m) \). Similarly \( \pi P \) is termed the predecessor of \( P \) and is given by the qbp \( \pi P = \{ \pi b; \overline{\pi b} \} \), where \( \pi b \) consists of all states \( S_1 \) such that \( S'_1 \in b \).

It follows that in a nonsingular flow table, every qbp has both a unique successor and a unique predecessor.

**Definition 7.2.** A qbp cycle, \( P_1, P_2, \ldots, P_k \), of length \( k \), is an ordered \( k \)-tuple of distinct qbp's such that \( \pi P_i = P_{i-1} \), for \( i = 2, 3, \ldots, k \), and \( \pi P_k = P_1 \).

It is clear that for any nonsingular flow table, all qbp's are contained in cycles. For example, consider the nonsingular machine \( M7.1 \) in Fig. 7.1.

<table>
<thead>
<tr>
<th>( S )</th>
<th>( S' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
</tr>
</tbody>
</table>

Fig. 7.1. Machine \( M7.1 \).
The qbp \( P_1 \) determined by \( b_1 = (A,B) \) is contained in the qbp cycle of length 5 given by, \( (A,B), (B,D), (D,E), (C,E), (A,C) \).

**Theorem 7.1.** If \( P_j \) is the qbp determined by any variable \( y_j \), \( j = 1, 2, \ldots, n \), in a minimal assignment for an autonomous linear sequential machine \( M \), which is nonsingular, the predecessor qbp \( \pi P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n \).

**Proof:** \( M \) can be characterized by an equation of the form,

\[
y^t = YA + B. \tag{7.1}
\]

If \( A = (a_{ij}) \) and \( B = (b_j) \), then

\[
y^t_j = y_1^t a_{1j} + y_2^t a_{2j} + \cdots + y_n^t a_{nj} + b_j. \tag{7.2}
\]

Since \( y_1, y_2, \ldots, y_n \) each determine a qbp at time \( t \), and \( y_1^t \) determines a qbp at time \( (t + 1) \) we have,

\[
\pi P_j = P_1 a_{1j} + P_2 a_{2j} + \cdots + P_n a_{nj} + b_j. \tag{7.3}
\]

Now \( b_j \) is a constant, therefore the predecessor of \( P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n \).

If \( \pi (\pi P) = \pi^2 P \), then we can say that \( \pi^k P \) is the \( k^{th} \) predecessor of \( P \), and we have the following:

**Corollary 7.1.** If \( P_j \) is defined as in Theorem 7.1, then \( \pi^k P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n \), for \( k \geq 1 \).

**Proof:** The case for \( k = 1 \) is proved by Theorem 7.1. For \( k = 2 \), from (7.1) we have

\[
y^{t+1} = YA^2 + BA + B \tag{7.4}
\]

If \( A^2 = (c_{ij}) \),

and \( BA + B = (d_j) \),
then
\[ y_j^{(t)} = (y_j^t)^t = y_1^t \cdot 1_j + y_2^t \cdot 2_j + \cdots + y_n^t \cdot n_j + d_j \] (7.5)
Again, \( y_1, y_2, \ldots, y_n \) each determine a qbp at time \( t \), and now \( y_j^{(t)} \)
determines a qbp at time \( (t + 2) \). Hence we have,
\[ \pi^2 P_j = P_1^t \cdot 1_j + P_2^t \cdot 2_j + \cdots + P_n^t \cdot n_j + d_j \] (7.6)
and \( \pi^2 P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n \). Clearly the
same argument holds for any \( k > 2 \).

From the definitions, it can be shown that, if \( P_1 \) and \( P_2 \)
are any two qbp's for a nonsingular autonomous machine, then
\[ \pi (P_1 + P_2) = \pi P_1 + \pi P_2 \cdot (7.7) \]
With repeated application of (7.7) it can be shown that,
\[ \pi (P_1 + P_2 + \cdots + P_m) = \pi P_1 + \pi P_2 + \cdots + \pi P_m \] (7.8)
and hence
\[ \pi^k (P_1 + P_2 + \cdots + P_m) = \pi^k P_1 + \pi^k P_2 + \cdots + \pi^k P_m \] (7.9)
where \( P_1, P_2, \ldots, P_m \) are qbp's for a nonsingular autonomous machine.

Definition 7.3. A valid qbp cycle of length \( k \) and order \( m \) is a
qbp cycle in which,

a) there are \( m \leq k \) mutually consistent and independent
qbp's.

b) the remaining \((k-m)\) qbp's are linear combinations of
the \( m \) independent qbp's.

For example, consider the nonsingular autonomous sequential
machine \( M \) in Fig. 7.2. A valid qbp cycle of length 7 and order 3
is given by \((A,C,F), (D,E,A), (E,F,C), (G,B,A), (F,G,D), (E,F,B),
(G,E,C), \) where \((A,C,F), (D,E,A), (E,F,B)\) are independent.
Lemma 7.2. In a valid qbp cycle of length $k$ and order $m$, any $m$ consecutive qbp's are independent.

Proof: The lemma is clearly satisfied for $k = m$. If $k > m$, then there exist $i > 1$ successive independent qbp's $P_1$, $P_{i-1}$, \ldots, $P_1$ such that $\pi P_j = P_{j+1}$, and $\pi P_i = a_1 P_i + a_{i-1} P_{i-1} + \ldots + a_1 P_1$. Therefore $\pi^2 P_i = a_1 \pi P_i + a_{i-1} P_i + \ldots + a_1 P_2$. Clearly $\pi^2 P_i$ is a linear combination of $P_i$, $P_{i-1}$, \ldots, $P_1$. Similarly it can be shown that each $\pi^r P_i$ is dependent until $r = k-1 + 1$, in which case we have $\pi^{k-1} P_i = P_i$. However, the qbp cycle is valid and of order $m$, therefore $i = m$, and any $m$ consecutive qbp's are independent.

Lemma 7.3. If $P_i$ can be used to define a variable $y_i$ in a minimal linear assignment, then $P_i$ is contained in a valid qbp cycle.

Proof: $P_i$ is contained in a qbp cycle, and $\pi P_i$ is a linear combination of $P_1$, $P_2$, \ldots, $P_n$, therefore $\pi P_i$ is either equal to $P_i$, or $\pi P_i$ and $P_i$ are independent. If $\pi P_i = P_i$, then $P_i$ is a valid qbp cycle of length 1 and order 1. If $\pi P_i$ and $P_i$ are independent then $\pi^2 P_i$, $\pi^3 P_i$, $P_i$ are either dependent or independent. If the former is true then $\pi^2 P_i$ is equal to either $P_i$ or $P_i + \pi P_i$, and in either case the lemma is satisfied for $\pi (P_i + \pi P_i) = P_i$. For the latter case we can repeat the argument until $\pi^r P_i = P_i$, and $P_i$ is contained in a valid qbp cycle.

Since we are only concerned with $m$ independent qbp's contained in a valid qbp cycle, and it may become rather laborious
to find the remainder, we make the following definition.

Definition 7.4. A qbp \( m \)-tuple consists of any \( m \) consecutive qbp's of a valid qbp cycle of order \( m \).

We now relate qbp \( m \)-tuples to linear sequential machines in the following two theorems. Linear sequential machines are divided into two disjoint sets, and the distinction between the sets is made by the characteristic matrix\(^{31}\) which can be either derogatory or nonderogatory. (An \((r \times r)\) matrix is said to be nonderogatory if the degree of its minimum polynomial\(^{31}\) is \( r \); otherwise it is derogatory.) We first consider the nonderogatory case.

Theorem 7.2. For every autonomous linear sequential machine \( M \), whose characteristic matrix is nonderogatory there exists a qbp \( n \)-tuple.

Proof: \( M \) can be characterized by the matrix equation,

\[
Y' = YA + B,
\]
where \( A \) is the transpose of the companion\(^{31}\) matrix, and \( B \) is a constant input vector. Therefore,

\[
y_1' = y_2 + b_1,
y_2' = y_3 + b_2,
\ldots
y_{n-1}' = y_n + b_{n-1},
y_n' = a_0y_1 + a_1y_2 + \ldots + a_{n-1}y_{n} + b_n.
\]
Since each \( y_1 \) determines a qbp \( P_1 \), and the addition of a constant leaves the partition unchanged we have,
\[ \pi P_1 = P_2, \]
\[ \pi P_2 = P_3, \]
\[ \ldots \]
\[ \pi P_{n-1} = P_n, \]
\[ \pi P_n = a_0 P_1 + a_1 P_2 + \ldots + a_{n-1} P_{n-1}. \]

Hence, \((P_n, P_{n-1}, \ldots, P_2, P_1)\) is a qbp n-tuple for \(M\).

To demonstrate this, consider the nonsingular autonomous sequential machine \(M7.2\) in Fig. 7.2. Here \(q = 7\), and \(n = 3\). A qbp 3-tuple for \(M\) is given by,

\[ b_1, b_2, b_3 = (A, B, D, F), (B, D, E, G), (B, G, F, G). \]

It can be shown that \(\pi b_1 = b_1 + b_3\); therefore, if we let \(y_1 = 1\) for \(S_j \in b_1\), we have the linear assignment for \(M7.2\) given in Fig. 7.3.

<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
</tr>
<tr>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
</tr>
</tbody>
</table>

Fig. 7.2. Machine \(M7.2\).
\[
\begin{array}{c|ccc}
 & y_1 & y_2 & y_3 \\
\hline
A & 1 & 0 & 0 \\
B & 1 & 1 & 1 \\
C & 0 & 0 & 1 \\
D & 1 & 1 & 0 \\
E & 0 & 1 & 0 \\
F & 1 & 0 & 1 \\
G & 0 & 1 & 1 \\
\end{array}
\]

Fig. 7.3. A linear assignment for M7.2.

The next state function for M7.2 are therefore,

\[y'_1 = y_1 + y_3,\]
\[y'_2 = y_1,\]
\[y'_3 = y_2.\]

The above theorem does not include autonomous linear sequential machines having a derogatory characteristic matrix, therefore for this class of machines we have the following theorem:

**Theorem 7.2.** For every autonomous linear sequential machine \( M \), whose characteristic matrix is derogatory, there exists \( r \) qbp \( n_i \)-tuples such that \( n_1 + n_2 + \ldots + n_r = n \) and the \( n \) qbp's constituting the \( n_i \)-tuples are mutually consistent and independent.

**Proof:** \( M \) can be characterized by the matrix equation

\[Y' = YA + B\]

where \( B \) is the constant input vector and the transpose of \( A \) is,
\[
A^T = \begin{pmatrix}
c_1 & 0 & 0 & \ldots & 0 \\
0 & c_2 & 0 & \ldots & 0 \\
0 & 0 & c_3 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & 0 & \ldots & c_r
\end{pmatrix}
\]

Each \(c_i\) is the companion matrix of the polynomial \(\Phi_i(\lambda)\). If the order of the \(i^{th}\) polynomial is \(n_i\), and we let
\[
N(i) = n_1 + n_2 + \ldots + n_{i-1},
\]
then
\[
y_{N(i) + 1} = y_{N(i) + 2} + b_{N(i) + 1}
\]
\[
y_{N(i) + 2} = y_{N(i) + 3} + b_{N(i) + 2}
\]
\[
\vdots
\]
\[
y_{N(i) + n_i - 1} = y_{N(i) + n_i} + b_{N(i) + n_i - 1}
\]
\[
y_{N(i) + n_1} = a_1 y_{N(i) + 1} + a_2 y_{N(i) + 2} + \ldots + a_{n_i - 1} y_{N(i) + n_i - 1} + b_{N(i) + n_i}
\]

Since each \(y_1\) determines a qbp \(P_1\) at time \(t\), and each \(y_j\) determines a qbp \(P_j\) at time \((t + 1)\) we have,
\[
\pi P_{N(i) + 1} = P_{N(i) + 2}
\]
\[
\pi P_{N(i) + 2} = P_{N(i) + 3}
\]
\[
\vdots
\]
\[ \pi P_N(i) + n_{1-1} = P_N(i) + n_1', \]
\[ \pi P_N(i) + n_1 = a_{10} P_N(i) + a_{11} P_N(i) + 1 + \ldots + a_{1n_1-1} P_N(i) + n_1. \]

Hence for \( M \) there exists the \( r \) qbp \( n_1 \)-tuples

\[
(P_N(i) + n_1', P_N(i) + n_1 - 1', \ldots, P_N(i) + 1)
\]

where \( i = 1, 2, \ldots, r \).

For example, consider the nonsingular autonomous machine \( M_7.3 \) in Fig. 7.4.

<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
</tr>
</tbody>
</table>

Fig. 7.4. Machine \( M_7.3 \).

We can show that for \( M_7.3 \) we have the qbp 1-tuple given by

\( b_1 = (A, E, F) \) and the qbp 2-tuple given by \( b_2, b_3 = (A, B, C, E), (A, C, D, F) \) such that \( P_1, P_2, P_3 \) are mutually consistent and independent.

We have shown that for every autonomous linear sequential machine there exists \( r > 0 \) qbp \( n_1 \)-tuples whose qbp's are mutually consistent and independent. However, only for a limited class of
machines which have a small number of states are we able to
exhaustively search for the required qbp \(n_1\)-tuples. We therefore
consider the output of the machine.

Lemma 7.4. Each binary output of an autonomous linear sequential
machine defines a qbp.

Proof: The output is given by the matrix equation

\[ Z = YC + D. \]

From this it is seen that any output \(z_i\) is given by,

\[ z_i = y_1 c_{i1} + y_2 c_{i2} + \cdots + y_n c_{in} + d_i. \]

Each \(y_j\) defines a qbp \(P_j\) such that the qbp's \(P_1, P_2, \ldots, P_n\) are
mutually consistent and independent. Hence, being a linear combi-
ination of \(P_1, P_2, \ldots, P_n\) and of the constant \(d_i\), \(z_i\) defines a
qbp.

Obviously, since each output qbp is a linear combination
of mutually consistent and independent qbp's, the output qbp's
must be mutually consistent.

Lemma 7.5. The output qbp \(P_j\) of an autonomous linear sequential
machine must be contained in a qbp \(k\)-tuple.

Proof: \(P_j\) is contained in a qbp cycle, and is a linear combination
of qbp's determined by the assigned variables. Therefore, by
considering the predecessors \(\pi_i P_j\) of \(P_j\) we can show that the
cycle is valid. Hence, \(P_j\) is contained in a qbp \(k\)-tuple.

As an example, consider the nonsingular autonomous
machine \(M7.4\) given by Fig. 7.5.
\[ S | S' | Z \\
A | E | 1 \\
B | F | 0 \\
C | B | 1 \\
D | C | 0 \\
E | G | 1 \\
F | A | 0 \\
G | D | 1 \\

Fig. 7.5. Machine M7.4.

We see that the output qbp given by \( b_1 = (A, C, E, G) \) is contained in the qbp 3-tuple determined by,

\[(A, C, E, G), (B, D, E, G), (C, D, F, G).\]

In summary, assuming that the machine satisfies the first necessary condition of linearity and is nonsingular, to find a minimal linear assignment for an autonomous sequential machine containing \( q \) states, we must find

\[ n = \lceil \log_2 q \rceil \text{ qbp's } (p_1, p_2, \ldots, p_n) \]

which are mutually consistent and independent, such that each predecessor qbp, \( \pi p_i \), and each output qbp is a linear combination of \( (p_1, p_2, \ldots, p_n) \). For a systematic procedure to find the \( n \) qbp's required, we take the following steps, which are illustrated in detail by the flow chart in Fig. 7.6:

a) Find the qbp determined by each output.

b) Determine the qbp \( n_1 \)-tuples containing the output qbp's.

c) Ensure that all qbp's contained in the \( n_1 \)-tuples are mutually consistent.

d) If the number of independent qbp's is less than \( n \),
Fig. 7.6. Flow chart of linearity procedure.
then find the qbp \( n_1 \)-tuple containing a qbp mutually consistent and independent with those already found, and revert to step c).

For example, consider the nonsingular autonomous machine N7.5 in Fig. 7.7.

<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>J</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>H</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>J</td>
<td>D</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 7.7. Machine N7.5.

We can easily show that the output qbp \( P_1 \), given by \( b_1 = (A, E, F, J) \), is contained in the qbp 4-tuple given by,

\[
b_1, b_2, b_3, b_4 = (A, E, F, J), (D, F, G, H), (G, E, G, I), (C, H, I, J)
\]

and that \( \pi b_1 = b_1 + b_2 + b_3 + b_4 \). Hence N7.5 is linear, and if we let \( y_1 = 1 \) for \( S_j \in b_1 \), we have the linear assignment functions for N7.5 in Fig. 7.8, and the following next state and output functions.
\[ y_1' = y_1 + y_2 + y_3 + y_4, \]
\[ y_2' = y_1, \]
\[ y_3' = y_2, \]
\[ y_4' = y_3, \]
\[ z = y_1. \]

<table>
<thead>
<tr>
<th></th>
<th>( y_1 )</th>
<th>( y_2 )</th>
<th>( y_3 )</th>
<th>( y_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>J</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 7.6. A linear assignment for M7.5.

As a further example, consider the nonsingular autonomous sequential machine M7.6 in Fig. 7.9, which, although actually not in reduced form, will be assumed to be so, for the purpose of illustration. The output qbp given by \( b_1 = (C, E, I) \) is contained in the qbp 2-tuple determined by

\[ b_1, b_2 = (C, E, I), (A, B, D). \]

Therefore, since \( q = 9, n = 4 \), and we require 4 qbp's which
<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>E</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>H</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>B</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 7.9. Machine M7.5.

satisfy the requirements, we must resort to an exhaustive search of all qbp's mutually consistent and independent with $b_1$, $b_2$.

It can be shown that one such qbp, determined by $b_3 = (A, I, F)$ is contained in the qbp 2-tuplo given by

$$b_3, b_4 = (A, I, F), (E, B, H),$$

and $b_1, b_2, b_3, b_4$ are mutually consistent and independent.
8. Nonautonomous Machines

In this section we consider only nonsingular machines; hence, each present state appears once and only once in each column of the flow table.

With regard to the input, two cases are considered:
In the first case the input \( r \)-tuple for each input is given; hence, the input \( q \times p \)'s are already specified. In the second case the inputs are labelled \( I_1, I_2, \ldots, I_p \); here \( r = \lceil \log_2 p \rceil \) input
\( q \times p \)'s that are mutually consistent and independent will have to be found. In the following definitions, lemmas and theorems no distinction between the two cases is necessary; however, both are illustrated, for the sake of added clarity, in the examples.

**Definition 8.1.** A total state \( S_1 I_j \) (or \( I_j S_1 \)) is an input state \( I_j \) and an internal state \( S_1 \).

**Definition 8.2.** The product of a set \( c \) of input states and a set \( b \) of internal states is denoted by \( bc \) and is defined as the set of all total states \( S_1 I_j \) such that \( I_j \in c \), and \( S_1 \in b \).

If \( S = \{ S_1, S_2, \ldots, S_q \} \) is the set of all internal states and \( I = \{ I_1, I_2, \ldots, I_p \} \) is the set of all input states, then \( IS = SI = \{ S_1 I_1, S_2 I_1, \ldots, S_q I_1 \} \) is the set of all total states.
Definition 8.1. The sum of a set \( c \) of input states and a set \( b \) of internal states is a set of total states defined by
\[
    b + c = b\overline{c} \lor \overline{b}c,
\]  
where \( \overline{b} \) is the complement of \( b \) with respect to \( S \), and \( \overline{c} \) is the complement of \( c \) with respect to \( I \).

For example, consider the flow table of machine \( M_{3.1} \) in Fig. 8.1.

<table>
<thead>
<tr>
<th>( S )</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A</td>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>E</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>D</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>E</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

Fig. 8.1. Machine \( M_{3.1} \).

If \( b = (A,3,2) \) and \( c = (1,2) \), then
\[
    bc = cb = (A_1,A_2,B_1,B_2,C_1,C_2),
\]
\[
    b + c = b\overline{c} \lor \overline{b}c = (A_3,B_3,C_3) \lor (D_1,D_2,E_1,E_2)
         = (A_3,B_3,C_3,D_1,D_2,E_1,E_2).
\]

Definition 8.4. The successor of a total state \( S_aI_b \) is the next internal state \( S_k \) for the present internal state \( S_a \) and present input \( I_b \), and is denoted by,
\[
    \sigma(S_aI_b) = S_k.
\]

Similarly, the predecessor of the internal state \( S_k \) is the set \( \beta \)
of all total states $S_{i,j}$ such that $\sigma(S_{i,j}) = S_k$, and is denoted

$$\pi S_k = \beta.$$ 

As an example, for $M8.1$

$$\sigma(D2) = A, \text{ and}$$

$$\pi(A) = (A_1, B_2, B_3, D_2).$$

**Definition 8.5.** The successor of a set of total states, is the union of successors of each total state. Similarly, the predecessor of a set of internal states is the union of predecessors of each internal state.

For example, consider $M8.1$ and $b = (A, B, C), c = (1, 2)$.

$$\sigma(bc) = (A, C, D, A, C, B) = (A, C, D, B).$$

$$\sigma(b + c) = (B, A, D, A, B, B) = (A, B, D, B).$$

$$\pi(b) = \pi(A) \cup \pi(B) \cup \pi(C)$$

$$= (A_1, B_2, B_3, D_2, E_2, A_3, B_2, C_1, A_2, B_3).$$

**Definition 8.6.** The sum of an input partition $Q = \{c; \overline{c}\}$ and a state partition $P = \{b; \overline{b}\}$ is defined as the total state partition

$$P + Q = \{b + c; \overline{b + c}\}. \quad (8.2)$$

**Definition 8.7.** The successor of $P + Q$ is defined as a grouping (or arrangement) of sets of internal states

$$\sigma(P + Q) = \{\sigma(b + c); \sigma(b + c)\}. \quad (8.3)$$
Obviously, $\sigma(P + Q)$ is an internal state qbp if and only if
$$\sigma(b + c) = \overline{\sigma(b + c)}.$$

**Definition 8.8.** The predecessor of an internal state qbp $P = \{ b; \overline{b} \}$ is defined as the total state partition
$$\pi P = \{ \pi b; \pi \overline{b} \}.$$  (8.4)

For example, consider the sequential machine $M_3.2$ given in Fig. 8.2.

<table>
<thead>
<tr>
<th>I</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
<td>D</td>
<td>G</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>C</td>
<td>E</td>
<td>H</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>G</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>A</td>
<td>H</td>
<td>E</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>F</td>
<td>B</td>
<td>G</td>
</tr>
<tr>
<td>F</td>
<td>E</td>
<td>H</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>E</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>B</td>
<td>F</td>
<td>D</td>
</tr>
<tr>
<td>I</td>
<td>K</td>
<td>L</td>
<td>J</td>
<td>I</td>
</tr>
<tr>
<td>J</td>
<td>I</td>
<td>J</td>
<td>L</td>
<td>K</td>
</tr>
<tr>
<td>K</td>
<td>J</td>
<td>I</td>
<td>K</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>K</td>
<td>I</td>
<td>J</td>
</tr>
</tbody>
</table>

**Fig. 8.2. Machine $M_3.2$.**

If $b = (A,C,F,G)$ and $c = (2,3)$, then it can be shown that:
\( \sigma(b + c) = (A, B, C, E, F, H, I, J, K, L) \).

**Definition 8.2.** An S/I pair, \( P/Q \), is an internal state qbp \( P \) and an input qbp \( Q \), such that \( \sigma(P + Q) \) is an internal state qbp.

It is clear that if \( \sigma(P_i + Q_j) = P_k \), then
\[ \pi P_k = P_i + Q_j. \]

As an example, consider machine MS1 given by Fig. 8.2.

It can be shown that the input and state qbp's \( Q \) and \( P \), given by \( c = (1, 2) \) and \( b = (A, B, D, E, I, L) \) respectively, constitute an S/I pair, for the successor of \( P + Q \) is an internal state qbp given by \( (A, C, D, F, K, L) \).

**Theorem 8.1.** If \( P_j \) is any qbp determined by any internal state variable \( y_j \), \( j = 1, 2, \ldots, n \), in a minimal assignment for a non-autonomous linear sequential machine \( M \), then \( \pi P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_r \), where \( Q_i \) is any qbp determined by the input variable \( x_i \), \( i = 1, 2, \ldots, r \).

**Proof:** \( M \) can be characterized by an equation of the form,
\[ y' = YA + XB. \]

If \( A = (a_{ij}) \), and \( B = (b_{ij}) \), then
\[ y'_j = y_{1j}a_1j + y_{2j}a_2j + \cdots + y_{nj}a_nj + x_{1j}b_1j + x_{2j}b_2j + \cdots + x_{rj}b_{rj}. \]

Since \( y_1, y_2, \ldots, y_n \) and \( x_1, x_2, \ldots, x_r \) each determine a qbp at time
t, and \( y_j \) determines a qbp at time \( t + 1 \), we have,

\[
\pi P_j = P_1a_{1j} + P_2a_{2j} + \ldots + P_na_{nj} + Q_1b_{1j} + Q_2b_{2j} + \ldots + Q_r b_{rj}.
\]

Hence, the predecessor of \( P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_r \).

It follows from the above theorem and the definition of an S/I pair, that the predecessor of any qbp defined by a state variable in a minimal assignment for a linear sequential machine is an S/I pair.

**Definition 6.10.** An S/I **cycle** of length \( k \) is an ordered \( k \)-tuple of S/I pairs \( P_1/Q_1, P_2/Q_2, \ldots, P_k/Q_k \) such that \( \pi P_i = P_{i-1} + Q_{i-1} \) for \( i = 2, 3, \ldots, k \) and \( \pi P_1 = P_k + Q_k \).

For example, consider MS.2 in Fig. 6.2. An S/I cycle of length 3 is given by \( b_1/c_1, b_2/c_2, b_3/c_3 = (B,E,G,H,I,J)/(1,4), (A,D,G,H,I,K)/(1,3), (C,F,G,H,J,K)/(1,2) \).

**Definition 6.11.** A **valid S/I cycle** of length \( k \) and order \( m \) is an S/I cycle in which

- **a)** there are \( m \leq k \) mutually consistent and independent internal state qbp's.
- **b)** each of the remaining \((k-m)\) state qbp's can be expressed as a linear combination of the \( m \) independent state qbp's.
- **c)** each of the \( k \) input qbp's can be expressed as a
linear combination of the r input qbp's which
are mutually consistent and independent.

As an example, it can be verified that the S/I cycle
for M3.2 given by \( b_1/c_1, b_2/c_2, b_3/c_3 \) above is valid and of order 3.
Similarly, it can be shown that the S/I cycle for M3.2 given by
\( b_4/c_4, b_5/c_5, b_6/c_6 \), where \( b_4, b_5, b_6 \) are given below, is valid
and of order 2.

\[ b_4, b_5, b_6 = (A, S, D, E, J, K), (A, G, D, F, I, J), (B, C, E, F, I, J). \]

Lemma 8.1. Any \( m \) consecutive internal state qbp's, in a valid
S/I cycle of length \( k \) and order \( m \), are independent.

Proof: The lemma is clearly satisfied for \( k = m \). If \( k > m \), then
starting with any state qbp \( P_i \) there exist \( i > 1 \) successive state
qbp's \( P_i, P_i - 1, \ldots, P_1 \) which are independent such that

\[ \pi P_j = P_j + 1 + Q_j + 1, \text{ for } j = 1, 2, \ldots, i - 1, \text{ and } \pi P_i = P_i + 1 + Q_i + 1, \text{ where } P_i + 1 = a_1 P_1 + a_2 P_2 + \ldots + a_i P_i \not\equiv P_i. \text{ Now, if } \pi P_i + 1 = P_i + 2 + Q_i + 2, \text{ then } P_i + 2 = a_1 a_i P_1 + (a_1 + a_1 a_2) P_2 + (a_2 + a_2 a_3) P_3 + \ldots + (a_i - 1 + a_i a_1) P_i, \text{ and clearly } P_i + 2 \text{ is a linear combination of } P_1, P_i - 1, \ldots, P_1. \]

The argument is repeated, until we have \( \pi P_k = P_1 + Q_1 \), each
state qbp \( P_j, j = 1, i + 1, \ldots, k \), being a linear combination
of \( P_1, P_2, \ldots, P_i \). However, the S/I cycle is valid and contains
\( m \) independent state qbp's, hence \( i = m \).
Lemma 8.2. Every state qbp $P_i$ defined by a variable $y_i$ in a minimal assignment for a nonautonomous linear sequential machine, which is nonsingular, is contained in a valid S/I cycle.

Proof: $\pi P_i = P_i^1 + Q_i^1$, where $P_i^1$ is a linear combination of $P_1, P_2, \ldots, P_n$. Therefore $P_i^1$ is either equal to $P_i$ or $P_i^1$ and $P_i$ are independent. If $P_i^1 = P_i$, then $P_i^1/Q_i^1$ is a valid S/I cycle of length 1 and order 1. If $P_i^1, P_i$ are independent then $\pi P_i^1 = P_i^2 + Q_i^2$, where $P_i^2$ is a linear combination of $P_1, P_2, \ldots, P_n$. Hence $P_i, P_i^1, P_i^2$ are either dependent or independent. If the former is true, then $P_i^2$ is equal to either $P_i$ or $P_i + P_i^1$, and in either case the lemma is satisfied for $\pi(P_i + P_i^1) = P_i + Q_i^1 + Q_i^2$. For the latter case we can repeat the argument until $\pi P_i^k = P_i + Q_i$. By a similar argument we can show that each $Q_i^k$ is a linear combination of $Q_1, Q_2, \ldots, Q_r$. Hence, $P_i$ is contained in a valid S/I cycle.

Again, as in the autonomous case, it may become a tedious chore to find all of the S/I pairs in an S/I cycle, when all that is necessary is to find $m$ consecutive S/I pairs whose $m$ state qbp's are mutually consistent and independent.

Definition 8.12. An S/I $m$-tuple consists of $m$ consecutive S/I pairs which are contained in a valid S/I cycle of order $m$.

As an example of an S/I $m$-tuple, consider $M_3, 12$ given in Fig. 8.2.
For the S/I 4-tuple given by $b_7/c_7$, $b_8/c_8$, $b_9/c_9$, $b_{10}/c_{10} = (A,B,F,H,I,K)/(1,5)$, $(A,E,F,G,J,K)/(1,2)$, $(D,E,F,H,I,J)/(1,4)$, $(G,D,E,G,I,K)/(1,5)$, it can be shown that $\pi F_7 = q_9 + P_7 + P_9 + P_{10}$.

We now relate S/I n-tuples to linear sequential machines in the following two theorems. The first deals with linear sequential machines having a nonderogatory characteristic matrix, while the second deals with machines having a derogatory characteristic matrix.

**Theorem 8.2.** For every nonautonomous linear sequential machine $M$, whose characteristic matrix is nonderogatory, there exists an S/I n-tuple.

**Proof:** $M$ can be characterized by an equation of the form

$$Y' = YA + XB,$$

where $A$ is the transpose of the companion matrix. Therefore, if $XB = (e_1)$ then

$$y'_1 = y_2 + e_0,$$

$$y'_2 = y_3 + e_2,$$

$$\cdots$$

$$y'_{n-1} = y_n + e_{n-1},$$

$$y'_n = y_1 a_0 + y_2 a_1 + \cdots + y_{n-1} a_{n-1} + e_n.$$ 

Now, each $y'_1$ determines a qbp of internal states and each $e_1$ determines a qbp which is a linear combination of the input qbp's. Hence, we have,
\[ \pi P_1 = P_2 + Q_1, \]
\[ \pi P_2 = P_3 + Q_2, \]
\[ \cdots \]
\[ \pi P_{n-1} = P_n + Q_{n-1}, \]
\[ \pi P_n = P_1 a_0 + P_2 a_1 + \cdots + P_n a_{n-1} + Q_n. \]

Therefore \((P_n/Q_{n-1}, P_{n-1}/Q_{n-2}, \ldots, P_1/Q_0)\) is an S/I n-tuple for \(N\).

**Theorem 8.3.** For every nonautonomous linear sequential machine \(N\), whose characteristic matrix is derogatory, there exist \(r\) S/I \(n_1\)-tuples such that \(n_1 + n_2 + \cdots + n_r = n\), and the \(n\) state qbp's contained in the \(n_1\)-tuples are mutually consistent and independent.

**Proof:** \(N\) can be characterized by the matrix equation,

\[ Y' = YA + XB, \]

where \(A\) is the sum of transposed companion matrices. It has been proved that for every companion matrix there exists an S/I n-tuple; hence for \(N\) there exists \(r\) S/I \(n_1\)-tuples such that \(n_1 + n_2 + \cdots + n_r = n\). The \(n\) state qbp's are mutually consistent and independent because the assignment is valid, which completes the proof of the theorem.

We have shown, in the two previous theorems, that for every nonautonomous linear sequential machine there exists \(k > 0\) S/I \(n_k\)-tuples such that \(n_1 + n_2 + \cdots + n_k = n\), and the \(n\) state qbp's are mutually consistent and independent and each of the
n input qbp's is a linear combination of r mutually consistent and independent input qbp's. The next problem to consider is that of finding the \( n_1 \)-tuples. As with the autonomous case, we first consider the output of the machine.

**Lemma 8.3.** Each binary output of a nonautonomous linear sequential machine defines a total state qbp, which is the sum of an input qbp and an internal state qbp.

**Proof:** The output is given by the matrix equation,

\[
Z = YO + XD.
\]

From this it is seen that any output \( z_1 \) is given by,

\[
z_1 = y_1^{c11} + y_2^{c21} + \cdots + y_n^{c_1} + x_1^{d11} + x_2^{d21} + \cdots + x_r^{d_1}.
\]

Each \( y_j \) defines a state qbp \( P_j \) such that \( P_1, P_2, \ldots, P_n \) are mutually consistent and independent. Similarly, each \( x_k \) defines an input qbp \( Q_k \) such that \( Q_1, Q_2, \ldots, Q_r \) are mutually consistent and independent. Hence, being a linear combination of \( P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_r \), \( z_1 \) defines a total state qbp which is the sum of an input qbp and an internal state qbp.

**Lemma 8.4.** The internal state qbp \( P_j \) defined by an output of a nonautonomous linear sequential machine, which is nonsingular, must be contained in an \( S/I \) \( k \)-tuple.

**Proof:** \( P_j \) is a linear combination of \( P_1, P_2, \ldots, P_n \). Each predecessor \( \pi P_i, i = 1, 2, \ldots, n \) is an \( S/I \) qbp, hence \( \pi P_j \) is an \( S/I \) qbp, say \( \pi P_j = P_j^1 + Q_j^1 \). Continuing the argument by considering
successive predecessors, it is easily shown that $P_j$ is contained in an S/I $k$-tuple.

In summary, assuming that the machine satisfies the first necessary condition of linearity and is nonsingular, to find a minimal linear assignment for a nonautonomous sequential machine containing $q$ internal states and $p$ inputs, we must find $n = \lceil \log_2 q \rceil$ internal state qbp's $(P_1, P_2, \ldots, P_n)$ which are mutually consistent and independent, and $r = \lceil \log_2 p \rceil$ input qbp's $(Q_1, Q_2, \ldots, Q_r)$ which are mutually consistent and independent, such that the predecessor of each state qbp, $\pi P$, and each output qbp is a linear combination of $(P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_r)$. For a systematic procedure to find the $n + r$ qbp's required, we take the following steps, which are illustrated in detail in Fig. 8.2:

a) Find the internal state and input qbp's, $P_j$ and $Q_j$, determined by each binary output.

b) Determine the S/I $n_1$-tuple which contains each $P_j$.

c) Ensure that all internal state qbp's contained in the $n_1$-tuples are mutually consistent, and that all input qbp's are mutually consistent.

d) If the number of independent state qbp's is less than $n$, then find the S/I $n_1$-tuple which contains a state qbp mutually consistent and independent with those already found, and revert to step c).
Fig. 9.2. Nonautonomous Linearity Procedure.
As an example, consider the nonautonomous sequential machine M3.2 given in Fig. 8.3.

<table>
<thead>
<tr>
<th>I</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>E/1</td>
<td>B/0</td>
<td>G/1</td>
<td>A/0</td>
<td>C/1</td>
<td>H/0</td>
<td>D/0</td>
</tr>
<tr>
<td>B</td>
<td>D/1</td>
<td>F/0</td>
<td>A/1</td>
<td>G/0</td>
<td>H/1</td>
<td>C/0</td>
<td>E/0</td>
</tr>
<tr>
<td>C</td>
<td>H/0</td>
<td>G/1</td>
<td>B/0</td>
<td>F/1</td>
<td>D/0</td>
<td>E/1</td>
<td>C/1</td>
</tr>
<tr>
<td>D</td>
<td>F/1</td>
<td>D/0</td>
<td>C/1</td>
<td>H/0</td>
<td>G/1</td>
<td>A/0</td>
<td>B/0</td>
</tr>
<tr>
<td>E</td>
<td>C/0</td>
<td>A/1</td>
<td>F/0</td>
<td>E/1</td>
<td>E/0</td>
<td>D/1</td>
<td>H/1</td>
</tr>
<tr>
<td>F</td>
<td>A/0</td>
<td>C/1</td>
<td>D/0</td>
<td>E/1</td>
<td>B/0</td>
<td>F/1</td>
<td>G/1</td>
</tr>
<tr>
<td>G</td>
<td>G/0</td>
<td>H/1</td>
<td>B/0</td>
<td>D/1</td>
<td>F/0</td>
<td>E/1</td>
<td>A/1</td>
</tr>
<tr>
<td>H</td>
<td>B/1</td>
<td>E/0</td>
<td>H/1</td>
<td>C/0</td>
<td>A/1</td>
<td>G/0</td>
<td>F/0</td>
</tr>
</tbody>
</table>

Fig. 8.3. Machine M3.2.

It can be easily shown that the output defines the state qbp $P_1$, and the input qbp $Q_4$ given by $(A, 3, D, H)$ and $(1, 3, 5)$ respectively, where $P_1$ is contained in the S/I 5-tuple given by $b_1/a_1, b_2/a_2, b_3/a_3 = (A, B, D, H)/(1, 2, 7), (B, D, E, F)/(1, 3, 6), (A, C, F, D)/(1, 4, 6)$. It is seen that $\pi P_1 = P_3 + Q_1 + Q_5$, and that $Q_4 = Q_1 + Q_5$. 
8.1 Nonautonomous Machines with Nonlinear Output

There are machines, that either do not have the output specified or the output is clearly nonlinear, in which case it is desirable to know whether the next state functions are linear or not. For the autonomous case, under the same circumstances, an exhaustive search was required; however, for a nonautonomous machine, while the search is still necessary, there is a method by which the search can be restricted. This method is presented in this section.

Definition 8.13. Two distinct states $S_i$ and $S_j$ will be called a primary corresponding pair (pop), denoted $[S_i; S_j]$, under inputs $I_a$ and $I_b$, if and only if there exists a state $S_k$, such that either, $\sigma(S_kI_a) = S_i$ and $\sigma(S_kI_b) = S_j$, or $\sigma(S_kI_a) = S_j$ and $\sigma(S_kI_b) = S_i$.

Obviously a pop is independent of the order in which the states are listed, i.e. $[S_i; S_j] = [S_j; S_i]$.

For example, consider the sequential machine given by the flow table of Fig. 8.4. For MS, the primary corresponding pairs under inputs 0 and 1 are:

$[A: D], [B: I], [C: K], [E: H], [F: L], [G: J]$. 
Lemma 6.5. For a nonmonotonic linear sequential machine (singular or nonsingular) two distinct pop's under inputs $I_a$ and $I_b$ are disjoint.

Proof: Let $M$ be a linear sequential machine with inputs $I_a$ and $I_b$ under the pop [$(S_a, S_b)$]. Where $S_a$ is the state of $S_a$ different from $S_b$ and $S_b$, exists such that $\sigma(S_a) = S_b$ and $\sigma(S_b) = S_a$.

\[ y_a = y_a A + y_b B, \]
\[ y_b = y_a A + y_b B. \]

By (6.5) and (6.6), the state $S_a$ is different from $S_b$, and therefore exists a state $S_c$ such that $\sigma(S_c) = S_b$. This means that $S_a$ and $S_b$ are different states.
that either \( [S_k; S_m] \) or \( [S_h; S_m] \) is a pcp under inputs \( I_a \) and \( I_b \). For the former, a state \( S_1 \) must exist, such that either,

a) \( Y_m = Y_1 A + X_a B \),
\( Y_k = Y_1 A + X_b B \),

or b) \( Y_m = Y_1 A + X_b B \),
\( Y_k = Y_1 A + X_a B \).

For a) we have,
\( Y_m = Y_k + X_b B + X_a B \).

Adding (8.5) we get
\( Y_m = Y_0 A + X_a B + X_b B + X_a B = Y_0 A + X_b B \).

From (8.6) \( Y_m = Y_h \), which contradicts the assumption that \( S_m \) was different from \( S_h \). The situations for both b) and \( [S_h; S_m] \) can be proved by symmetry, hence two distinct pcp's under inputs \( I_a \) and \( I_b \) are disjoint.

**Lemma 8.6.** For every pair of distinct inputs \( I_a \) and \( I_b \), in a flow table of a nonsingular linear sequential machine \( H \) containing \( q \) states, there exists \( q/2 \) pcp's.

**Proof:** There are \( q \) pcp's \( \{ \sigma (S_i I_a); \sigma (S_i I_b) \} \) for \( i = 1,2,\ldots, q \); however, \( H \) is nonsingular hence \( \sigma (S_i I_a) = \sigma (S_i I_b) = S \) and each of the \( q \) states appears twice. Since two distinct pcp's are disjoint, there are exactly \( q/2 \) pcp's.

Obviously, from the above lemma, \( q \) is necessarily an even number for a nonautonomous linear sequential machine, which is nonsingular and fully specified.
It is also a consequence of the lemma that for any $S_i$ of a nonsingular machine and any two inputs $I_a$ and $I_b$ the states $\sigma(S_iI_a) = S_g$ and $\sigma(S_iI_b) = S_h$ are distinct internal states, and that there exists a state $S_j$ such that $\sigma(S_jI_a) = S_h$ and $\sigma(S_jI_b) = S_g$; i.e. the flow table has this pattern

<table>
<thead>
<tr>
<th></th>
<th>$I_a$</th>
<th>...</th>
<th>$I_b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_i$</td>
<td>$S_g$</td>
<td>...</td>
<td>$S_h$</td>
</tr>
<tr>
<td>$S_j$</td>
<td>$S_h$</td>
<td>...</td>
<td>$S_g$</td>
</tr>
</tbody>
</table>

(see for example Fig. 8.2). This leads to the following definition:

**Definition 8.14.** Two states $S_g$ and $S_h$ are called a secondary corresponding pair, (scp) denoted $<S_g; S_h>$ under inputs $I_a$ and $I_b$, if and only if there exists a pop $[S_i; S_j]$ under $I_a$ and $I_b$ such that $\sigma(S_gI_a) = \sigma(S_hI_b) = S_i$ and $\sigma(S_gI_b) = \sigma(S_hI_a) = S_j$.

For example, consider $N8.3$ given by Fig. 8.4. It is easily shown that the scp's for $N8.3$ are:

$<A; J>$, $<B; F>$, $<C; H>$, $<D; G>$, $<E; K>$, $<I; L>$.  

Clearly, for a nonsingular machine all scp's are disjoint.

It also follows from Lemma 8.7 and Definition 8.9 that for every pair of distinct inputs $I_a$ and $I_b$ of a nonsingular linear sequential machine with $q$ states, there are $q/2$ scp's.

Having defined corresponding pairs (cp's) and proved some of their elementary properties, we can now state their use in finding a linear assignment.
Theorem 8.4. For every linear sequential machine $M$, and any two distinct inputs $I_a$ and $I_b$, there exists a qbp, $P_c$, which can be used to define a variable $y_c$ in a valid linear assignment for $M$, which separates all qp's under $I_a$ and $I_b$.

Proof: To prove that $P_c$ separates all qp's, it is necessary and sufficient to prove the existence of qbp's $P_j$ and $P_k$ such that

$$\pi P_c = P_j + Q, \quad (8.7)$$
$$\pi P_k = P_c + Q, \quad (8.8)$$

where $P_j$ and $P_k$ are linear combinations, not necessarily distinct, of $P_1, P_2, \ldots, P_n$, and $Q$ is any input qbp which separates $I_a$ and $I_b$.

For the inputs $I_a$ and $I_b$, $M$ can be characterized by an equation of the form

$$Y' = YA + XB, \quad (8.9)$$

where $X = (x \mid 1)$ is the input vector, having the binary variable $x$ and a constant input $1$, and $A$ is in canonical form. Since $A$ is either a companion matrix or a sum of companion matrices (rational canonical form) we need only consider the companion matrix case.

We therefore assume $A$ to be of the form

$$A = \begin{pmatrix}
0 & 1 & 0 & 0 & \cdots & 0 \\
0 & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 0 & 1 & \cdots & 0 \\
& \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & 0 & 0 & \ddots & 1 \\
& & a_0 & a_1 & a_2 & \cdots & a_{n-1}
\end{pmatrix}. \quad (8.10)$$

However, $A$ is nonsingular, therefore $a_0 = 1$. $M$ is nonautonomous,
Theorem 8.4. For every linear sequential machine \( M \), and any two distinct inputs \( I_a \) and \( I_b \), there exists a qbp, \( P_o \), which can be used to define a variable \( y_o \) in a valid linear assignment for \( M \), which separates all cp's under \( I_a \) and \( I_b \).

Proof: To prove that \( P_o \) separates all cp's, it is necessary and sufficient to prove the existence of qbp's \( P_j \) and \( P_k \) such that

\[
\pi P_o = P_j + Q, \tag{8.7}
\]
\[
\pi P_k = P_o + Q, \tag{8.8}
\]

where \( P_j \) and \( P_k \) are linear combinations, not necessarily distinct, of \( P_1, P_2, \ldots, P_n \), and \( Q \) is any input qbp which separates \( I_a \) and \( I_b \).

For the inputs \( I_a \) and \( I_b \), \( M \) can be characterized by an equation of the form

\[
Y' = YA + XB, \tag{8.9}
\]

where \( X = \begin{pmatrix} x \\ 1 \end{pmatrix} \) is the input vector, having the binary variable \( x \) and a constant input \( 1 \), and \( A \) is in canonical form. Since \( A \) is either a companion matrix or a sum of companion matrices (rational canonical form) we need only consider the companion matrix case. We therefore assume \( A \) to be of the form

\[
A = \begin{pmatrix}
0 & 1 & 0 & 0 & \ldots & 0 \\
0 & 0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 0 & 1 & \ldots & 0 \\
0 & 0 & 0 & 0 & \ldots & 1 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & \ldots & 1 \\
\end{pmatrix}. \tag{8.10}
\]

However, \( A \) is nonsingular, therefore \( a_0 = 1 \). \( M \) is nonautonomous,
hence there exists at least one $b_1 = 1$, in the matrix

$$B = \begin{pmatrix} b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & \cdots & c_n \end{pmatrix}. \quad (8.11)$$

There are four cases to consider: If $b_1 = 0$, then there exists a $b_i = 1$, such that $b_i - 1 = 0$. The case where $b_i + 1 = 0$ is treated in a), and where $b_i + 1 = 1$ is treated in b). If $b_i = 1$, and $b_n = 1$ is shown in c), and where $b_n = 0$ is shown in d).

a) $b_1 = 0$, $b_1 - 1 = 0$, $b_i = 1$, $b_i + 1 = 0$.

From $(8.9)$, $(8.10)$ and $(8.11)$ we have:

$$y_i^1 = y_n + c_1,$$
$$y_i^1 - 1 = y_i - 2 + a_i - 2y_n + c_i - 1,$$
$$y_i^1 = y_i - 1 + a_i - 1y_n + x + c_i,$$
$$y_i^1 + 1 = y_i + a_i y_n + c_i + 1.$$

Either,

1) $a_i = a_i - 1$, in which case

$$y_i^1 - 1 + y_i^1 = y_i - 2 + y_i - 1 + a_i - 2y_n + a_i - 1y_n + x + c_i - 1 + c_i,$$
$$y_i^1 + y_i^1 + 1 = y_i - 1 + y_i + x + c_i + c_i + 1.$$

Hence, $P_0$ defined by $y_i - 1 + y_i$, $P_j$ defined by $y_i - 2 + y_i - 1 + a_i - 2y_n + a_i - 1y_n$ and $P_k$ defined by $y_i + y_i + 1$ satisfy the requirements of $(8.7)$ and $(8.8)$. 
or,

ii) \( a_i \neq a_{i-1} \), in which case,

\[
y_{i-1}^t + y_i^t = y_{i-2}^t + y_{i-1}^t + a_{i-2}Y_n + a_{i-1}Y_n + x + c_{i-1} + c_i,
\]

\[
y_i^t + y_i^t + y_i^t + 1 = y_{i-1}^t + y_i^t + x + c_{i-1} + c_i + c_{i+1} + 1.
\]

Hence, \( P_c \) defined by \( y_{i-1}^t + y_i^t \), \( P_j \) defined by \( y_{i-2}^t + y_{i-1}^t + a_{i-2}Y_n + a_{i-1}Y_n \) and \( P_k \) defined by \( y_i^t + y_i^t + y_i^t + 1 \) satisfy the requirements of (8.7) and (8.8).

b) \( b_1 = 0, b_{i-1} = 0, b_i = 1, b_i + 1 = 1 \).

From (8.9), (8.10) and (8.11) we have:

\[
y_1^t = y_n + c_1,
\]

\[
y_{i-1}^t = y_{i-2}^t + a_{i-2}Y_n + c_{i-1} + c_i,
\]

\[
y_i^t = y_{i-1}^t + a_{i-1}Y_n + x + c_i,
\]

\[
y_i^t + 1 = y_i^t + a_iY_n + x + c_i + 1.
\]

Either,

i) \( a_i = 0 \), in which case,

\[
y_i^t = y_{i-1}^t + a_{i-1}Y_n + x + c_i,
\]

\[
y_i^t + 1 = y_i^t + x + c_i + 1.
\]

Hence, \( P_c \) defined by \( y_1^t \), \( P_j \) defined by \( y_{i-1}^t + a_{i-1}Y_n \) and \( P_k \) defined by \( y_i^t + 1 \) satisfy the requirements of (8.7) and (8.8).
or,

ii) \( a_1 = 1 \), in which case,

\[
\begin{align*}
    y'_i &= y_{i-1} + a_{i-1}y_n + x + c_i, \\
    y'_i + y'_i + 1 &= y_i + x + a_i + c_i + 1.
\end{align*}
\]

Hence, \( P_o \) defined by \( y'_i \), \( P_j \) defined by \( y_{i-1} + a_{i-1}y_n \), and \( P_k \) defined by \( y_i + y_i + 1 \) satisfy the requirements of (8.7) and (8.8).

do) \( b_1 = 1, b_n = 1 \).

From (8.9), (8.10) and (8.11) we have:

\[
\begin{align*}
    y'_1 &= y_n + x + c_1, \\
    y'_n &= y_{n-1} + a_{n-1}y_n + x + c_n.
\end{align*}
\]

Hence \( P_o \) defined by \( y'_n \), \( P_j \) defined by \( y_{n-1} + a_{n-1}y_n \) and \( P_k \) defined by \( y_1 \) satisfy the requirements of (8.7) and (8.8).

d) \( b_1 = 1, b_n = 0 \).

Hence there exists a \( b_1 = 0 \), such that \( b_{1-1} = 1 \).

From (8.9), (8.10) and (8.11) we have:

\[
\begin{align*}
    y'_1 &= y_n + x + c_1, \\
    y'_{i-1} &= y_{i-2} + a_{i-2}y_n + x + c_{i-1}, \\
    y'_i &= y_{i-1} + a_{i-1}y_n + c_i, \\
    y'_n &= y_{n-1} + a_{n-1}y_n + c_n.
\end{align*}
\]
Either,

i) $a_{i-1} = 1$, in which case,

$$y_{i-1}' = y_{i-2} + a_{i-2}y_n + x + c_{i-1},$$
$$y_1' + y_1' = y_{i-1} + x + c_1 + c_1.'$$

Hence, $P_0$ defined by $y_{i-1}$, $P_j$ defined by

$y_{i-2} + a_{i-2}y_n$ and $P_k$ defined by $y_1 + y_1$

satisfy the requirements of (8.7) and (8.8).

or,

ii) $a_{i-1} = 0$, in which case,

$$y_{i-1}' + y_1' = y_{i-2} + y_{n-1} + a_{i-2}y_n +$$
$$a_{i-1}y_n + x + c_{i-1} + c_n,$$

$$y_1' + y_1' = y_n + y_{i-1} + x + c_1 + c_1.$$

Hence, $P_0$ defined by $y_{i-1} + y_n$, $P_j$ defined

by $y_{i-2} + y_{n-1} + a_{i-2}y_n + a_{i-1}y_n$ and $P_k$

defined by $y_1 + y_1$ satisfy the requirements

of (8.7) and (8.8).

This completes the proof of the theorem.

Since $P_0$ is a linear combination of $P_1, P_2, \ldots, P_n$;

it can be shown, by an argument similar to the proof of Lemma 8.4,

that $P_0$ is contained in an S/I $n_1$-tuple.

As an example, consider $M8.3$ and the associated op's:
\[
\begin{array}{cc}
[A; D] & <A; J> \\
[B; I] & <B; F> \\
[C; K] & <C; H> \\
[D; G] & <D; G> \\
[E; L] & <E; K> \\
[G; J] & <I; L>
\end{array}
\]

It is easily shown that the only qbp's which separate all cp's are given by:

\[b_1, b_2, b_3, b_4 = (A, B, C, E, G, L), (A, B, G, H, K, L), (A, C, E, F, G, I),\]
\[(A, P, G, H, I, K).\]

Furthermore, it can be shown that \(P_1\) and \(P_2\) are contained in the S/I 2-tuples given by: \(b_1/c, b_2/I\) and \(b_3/c, b_6/I\), where

\[c, b_5, b_6 = (1), (A, B, C, F, H, J), (A, C, H, I, J, L),\]

\[\pi P_1 = Q + P_1 + P_5,\]
\[\pi P_2 = Q + P_2 + P_6.\]

It is worth noting that \(b_3 = b_1 + b_5 + b_6\) and \(b_4 = b_5 + b_2 + b_6\).

Therefore, in order to show that an autonomous sequential machine, which satisfies the first necessary condition of linearity, is linear or not, we take the following steps, which are illustrated in detail by the flow chart in Fig. 8.5:

a) Select two inputs \(I_a\) and \(I_b\).

b) Find all cp's for \(I_a\) and \(I_b\).

c) Find a distinct qbp of states \(P\) separating all cp's.
Fig. 8.5. Linearity procedure for nonlinear output.
d) Is P contained in an S/I \( n_1 \)-tuple? If the answer is 'no' return to c), otherwise continue to e).

e) If the number of independent state qbp's is less than \( n \), then find the S/I \( n_1 \)-tuple containing a state qbp mutually consistent and independent with those already found.

f) Ensure that all state qbp's contained in the \( n_1 \)-tuples are mutually consistent and that all input qbp's are mutually consistent and revert to e).

For example, consider machine MS.4 shown in Fig. 8.6. The input qbp's \( Q_1 \), \( Q_2 \), \( Q_3 \) for MS.4 are given by \( a_1 = 3 \), \( a_2 = 4 \), \( a_3 = 5 \), \( b_1 = 2 \), \( b_2 = 3 \), \( b_3 = 4 \), where 1, 2, 3, 4, 5 = 000, 011, 101, 110, 111. From 1 and 2 we have the following cp's:

\[
[A: G], [B: H], [C: O], [D: N], [E: I], [F: X],
[J: K], [L: P], <A: N>, <B: O>, <C: G>, <D: H>,
\]

It is easily shown that the qbp, which separates all cp's given by \((A,B,D,E,F,J,O,P)\) is not contained in an S/I \( n_1 \)-tuple; however, the qbp \( P_1 \) given by \( b_1 = (A,C,D,E,F,J,O,P) \) is contained in the S/I 4-tuple given by: \( P_1/Q_1 + Q_3 \), \( P_2/Q_1 + Q_2 \), \( P_3/Q_2 + Q_3 \), \( P_4/Q_2 \), where \( \pi P_1 = P_1 + P_2 + P_4 + Q_2 \) and \( \pi P_2, P_3, P_4 \) are given by \( b_2 = (A,B,D,H,I,J,O,P) \), \( b_3, b_4 = (A,B,D,H,I,J,O,P) \), \( b_4 = (A,B,D,H,I,J,L) \), \( (C,D,F,H,L,N,O,P) \).
Fig. 8.6. Machine No. 4.
9. Singular Machines

If the machine satisfies the first necessary condition of linearity and is singular, then $e < n$ and $d < q$, (see Theorem 6.1). We first consider the singular autonomous case.

9.1 The Autonomous Case

Definition 9.1. The nonsingular states of a singular autonomous sequential machine consist of all states which appear in cycles, and all other states are termed transient states.

The first problem of finding a linear assignment for a singular autonomous machine is that of finding a linear assignment for the nonsingular states using the methods presented in Section 7. Once the required $k > 0$ qbp $n_1$-tuples have been found for the nonsingular states, we can add the transient states to the qbp's so that, for the resulting qbp's, $\pi b_j$ contains the predecessors of $b_j$; remembering that in a singular machine each state does not necessarily have a unique predecessor.

We are now faced with the problem of finding an assignment for what we will call the transient variables, or in terms of partitions, transient qbp's. Because there is very little theory existing in the references about singular linear sequential machines, the search for the transient qbp's will be demonstrated by an example.

Since $e < n$, we can find $(n - e)$ qbp's, $P_1, P_2, \ldots, P_{n-e}$.
satisfying our consistence and independence requirement, such that \( \pi P_1 = 0 \), where \( \pi = \sum_0^d \pi_0 \). These qbp's are based on present states which do not appear as next states. If the number \( f \) of nonsingular states is less than the number of distinct next states, \( f < d \), we must find \( t = \lfloor \log_2 (d - f) \rfloor \) qbp's, \( P_1^*, P_2^*, \ldots, P_t^* \), again satisfying our consistence and independence requirement, such that each predecessor \( \pi P_1^* \) is a linear combination of \( (P_1, P_2, \ldots, P_{n-e}, P_1^*, P_2^*, \ldots, P_t^*) \).

For example, consider the autonomous machine M9.1 given in Fig. 9.1.

<table>
<thead>
<tr>
<th>S</th>
<th>S'</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>F</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
</tr>
<tr>
<td>G</td>
<td>E</td>
</tr>
<tr>
<td>H</td>
<td>D</td>
</tr>
<tr>
<td>I</td>
<td>C</td>
</tr>
<tr>
<td>J</td>
<td>D</td>
</tr>
</tbody>
</table>

Fig. 9.1. Machine M9.1.

It can be shown that M9.1 satisfies the first necessary condition of linearity and is singular. The only states which
appear in cycles are A, B and F, hence these are the nonsingular states of the machine. A qbq cycle of length 3 and order 2 is given by \( b_1, b_2, b_3 = (A), (F), (B) \), where \( b_3 = b_1 + b_2 + S \). Since states E and D are both predecessors of A, states B and D must be in the same block. Similarly, F, C, H and J are the predecessors of (B,D) and we have the block (F, C, H, J). By similar arguments we can show that the qbq cycle of length 3 and order 2, which includes all states is given by \( b_1, b_2, b_3 = (A, E, I), (F, C, H, J), (B, D, G) \), and again \( b_3 = b_1 + b_2 + S \).

We now need the necessary transient qbq's. From the flow table we see that states G, H, I and J do not have a predecessor.

Hence, if \( b_4 = (G, H, I, J) \), then \( \pi b_4 = \emptyset \) and \( \pi P_4 = 0 \).

We now note that there are 3 intermediate states C, D and E such that if \( b_5 = (C, D, E) \) then \( \pi b_5 = b_4 \). However, \( b_1 \) and \( b_5 \) are not consistent, because \( \#(b_1 b_5) = 5 \), but we can overcome this by adding state H to \( b_5 \). We can now show that \( P_1, P_2, P_4, P_5 \), given by \( b_1, b_2, b_4, b_5 = (A, E, I), (F, C, H, J), (G, H, I, J), (C, D, E, H) \) are mutually consistent and independent, such that each predecessor is a linear combination of \( P_1, P_2, P_4, P_5 \).

If we assign \( y_1 = 1 \) for \( b_1 \in b_1 \) the assignment given in Fig. 9.2 results. The next state functions for \( y_9, i \) are given by,

\[
\begin{align*}
y_1' &= y_1 + y_2 + 1, \\
y_2' &= y_1, \\
y_4' &= 0, \\
y_5' &= y_4.
\end{align*}
\]
\[
\begin{array}{c|cccc}
  & v_1 & v_2 & v_4 & v_5 \\
\hline
A & 1 & 0 & 0 & 0 \\
B & 0 & 0 & 0 & 0 \\
C & 0 & 1 & 0 & 1 \\
D & 0 & 0 & 0 & 1 \\
E & 1 & 0 & 0 & 1 \\
F & 0 & 1 & 0 & 0 \\
G & 0 & 0 & 1 & 0 \\
H & 0 & 1 & 1 & 0 \\
I & 1 & 0 & 1 & 0 \\
J & 0 & 1 & 1 & 1 \\
\end{array}
\]

Fig. 9.2. A linear assignment for M9.1.

We can show that, as for a nonsingular machine, each output must be a qbp and the output qbp's must be contained in qbp \( n_1 \)-tuples unless they are transient outputs, in which case they can be used to determine transient qbp's.

2.2 The Nonsynchronous Case

The assignment problem for a singular nonsynchronous machine is slightly different to that of a singular autonomous machine. However, we can use a combination of techniques used in the previous cases with the exception of scp's. For example, consider machine M9.2 given in Fig. 9.3. Under inputs 1 and 2, there are no scp's and under inputs 2 and 3 the scp's are not distinct.
Fig. 9.3. Machine M9.2.

Again, we demonstrate the techniques by example.

For M9.2 we see that the predecessor of the qbp $P_1$ given by $b_1 = (A, C, D, E)$ is given by $\pi P_1 = Q_1$, where $Q_1$ is given by $c_1 = (1)$. Under inputs 1 and 2 we have the following pcp's:

\[
\begin{array}{c|ccc}
  I & 1 & 2 & 3 \\
  \hline
  A & D & B & F \\
  B & G & H & E \\
  C & C & E & H \\
  D & D & B & F \\
  E & A & F & B \\
  F & G & H & E \\
  G & C & E & H \\
  H & A & F & B \\
\end{array}
\]

\[\begin{array}{l}
[A; F], [B; D], [C; E], [G; H]. \quad \text{We note that there are only 3 internal state qbp's which separate all pcp's and are mutually consistent with } P_1. \quad \text{These are } P_2, P_3 \text{ and } P_4 \text{ given by } b_2, b_3, b_4 = (A, B, C, H), (A, B, E, G), (A, D, E, H) \text{ respectively. It is easily shown that}
\end{array}\]

\[
\begin{align*}
\pi P_2 &= P_2 + P_3 + Q_2, \\
\pi P_3 &= P_1 + Q_2, \\
\pi P_4 &= P_4 + Q_1,
\end{align*}
\]
where \( a_2 \) is given by \( a_2 = (2) \). Hence any three independent qbp's selected from \( P_1, P_2, P_3, P_4 \) will provide a valid assignment.
10. Conclusions and Recommendations for Further Research

In conclusion, a method has been presented by which it can be determined whether a restricted class of sequential machines are linearly realizable with a minimum number of delay elements. Linear sequential machines have been introduced and some of their basic properties have been examined including the canonical forms and the transfer function. A method has been presented for finding the transfer function using signal flow graphs. The problems involved in determining from the flow table whether a sequential machine is linearly realizable are presented. Several approaches to the problem were considered, and the conclusion drawn that none of the approaches seem to yield the desired solution. We have therefore examined the properties and behaviour of a class of binary partitions called qbp's. It has been shown that the qbp approach is straight-forward and in general provides a ready solution. In the case of a flow table containing a large number of states, it may be argued that the method is too cumbersome and unwieldy; however, it is felt that this drawback can be readily overcome since the method can be programmed on a digital computer.

There are, however, some problems which were considered to be beyond the scope of this thesis, as a result, these are presented as recommendations for further research:
1) In section 9 it was indicated that the qbp approach can be used for singular machines. With some further research into the structure and behaviour of singular machines it is felt that an algorithmic method could be found for determining whether a singular machine is linear or not.

2) It should be possible to devise a method for reducing the search problem for nonautonomous machines when the output is either not specified or nonlinear.

3) Machines whose flow tables are not fully specified should be considered.

4) It is felt that by using qbp's one could determine the amount of nonlinearity (in number of 'and' gates) for a nonlinear sequential machine.

5) The method of qbp's could be applied to the assignment problem in general.
BIBLIOGRAPHY


VITA

Name: Wayne Alton Davis

Born: 16 November 1931, Fort Macleod, Alberta.

Educated:

Primary: Fort Macleod, Alberta.

Secondary: Fort Macleod, Alberta.

University: The George Washington University Washington, D.C.

Degree: Bachelor of Science in Engineering (B.S.E.)

Option: Mathematics