Sequential machines

Marcel Pilate

Atlanta University
SEQUENTIAL MACHINES

A THESIS
SUBMITTED TO THE FACULTY OF ATLANTA UNIVERSITY IN
PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

BY
MARCEL PILATE

DEPARTMENT OF MATHEMATICS

ATLANTA, GEORGIA
AUGUST 1968
ACKNOWLEDGMENTS

Some of the most profound works done on the mathematical theory of sequential machines are those of Hartmanis and Stearns. Their works were the most influential in my choice and arrangement of materials in this thesis. I owe special thanks to Dr. Williams for his helpful discussions on this subject which have resulted, it is hoped, in a far better thesis than would have otherwise been possible.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS .............................................. ii

Chapter

I. FUNDAMENTAL CONCEPTS ........................................ 1

- Introduction
- Preliminary Mathematical Concepts
- Partial Ordering and Lattices

II. MACHINES AND STATE ASSIGNMENT PROBLEM ............... 6

- Introduction
- The Machine and State Assignment
- Partitions and State Assignment with Substitution Property
- Computation of S.P. Partitions
- Further Explanation Through Selected Problems

III. PAIR ALGEBRA AND DECOMPOSITION ....................... 25

- Pair Algebra
- Decomposition

BIBLIOGRAPHY .................................................. 32
CHAPTER I

FUNDAMENTAL CONCEPTS

Introduction

This thesis is primarily concerned with sequential machines and the mathematical concepts that govern them. A very active part of mathematics is applied to the study of sequential machines which are abstract models of digital computers.

Since the mid 1950's, the amount of research conducted in the area of sequential machines has increased very rapidly. The initial development in the design of information-processing systems dealt mainly with the problems of hardware design. As the hardware problems were solved, men like D.A. Huffman and E.T. Moore begin to develop theoretical techniques necessary for sequential machines to be placed on sound abstract mathematical bases. We shall discuss this organized body of techniques and results which deals with the problems of how sequential machines can be realized from sets of smaller component machines and how these component machines have to be interconnected for the desired and controlled information flow.

Preliminary Mathematical Concepts

In this section we state some standard mathematical concepts which may have some peculiarity in their applications in sequential machine theory. The writer assumes that the reader has a thorough knowledge of set theory, relations, and functions. Although most of the concepts from abstract algebra and Boolean Algebra will be discussed as needed, the
reader is expected to have some knowledge of them.

A basic concept in our topic is the idea of partitions on sets and some special means of combining them; therefore we start by looking at a set $s$.

**Definition 1**: Let $P = \{B_1, B_2, \ldots, B_n\}$ be a finite collection of subsets of $S$. The set $P$ is called a partition of $S$ if and only if

1. $\bigcup_{i=1}^{n} B_i = S$
2. $B_i \cap B_j = \emptyset$ if $i \neq j$.

Each of the distinct subsets $B_i$ is called a block of the partition $P$. When we write out a partition, we distinguish blocks with bars and semicolons. For example, if $S = \{a, b, c, d, e, f, g\}$ and a partition $P$ on $S$ has blocks $\{a, b, c\}$, $d$, and $\{e, f, g\}$, we write $P = \{\overline{a, b, c} \; d; \; \overline{e, f, g}\}$.

We say that $S = t(P)$ if and only if $s$ and $t$ are contained in the same block $P$. In the above example $e \equiv t(P)$. If $R$ is an equivalence relation on $S$, then the set of equivalence classes defines a partition $P$ on $S$ and conversely. That is, if $R$ defines $P$, then $s \equiv t$ if and only if $S = t(P)$.

**Definition 2**: If $P_1$ and $P_2$ are partitions on $S$, then the product $P_1 \cdot P_2$ is the partition on $S$ such that $s \equiv t(P_1 \cdot P_2)$ if and only if $s = t(P_1)$ and $s = t(P_2)$.

**Definition 3**: If $P_1$ and $P_2$ are partitions on $S$, then the sum $P_1 + P_2$ is the partition on $S$ such that $s \equiv t(P_1 + P_2)$ if and only if there exists a sequence in $S$, $s = s_0, s_1, \ldots, s_n = t$ for which either $s_i = s_{i+1}(P_1)$ or $s_i \equiv s_{i+1}(P_2)$, $0 \leq i \leq n-1$.

$P_1 \cdot P_2$ is obtained by intersecting blocks of $P_1$ and $P_2$,

$$B_{P_1 \cdot P_2}(s) = B_{P_1}(s) \cap B_{P_2}(s).$$
To obtain $B_{p_1 + p_2}(s)$, we proceed inductively.

Let $B_1(s) = B_{p_1}(s) \cup B_{p_2}(s)$ and for $i > 1$ let $B_{i+1}(s) = B_i(s) \cup \{B_i : B_i \text{ is a block of } P_1 \text{ or } P_2\}$ and $B_{i+1}(s) = \emptyset$.

**Example:** Let $S = \{a, b, c, d, e, f, g, h\}$ and $P_1 = \{a, b; c, d; e, f, g, h\}$

$P_2 = \{a, d; b, c; e, f, g, h\}$

Then $P_1 \cdot P_2 = \{a; b; c; d; e, f, g, h\}$

and $P_1 + P_2 = \{a, b, c, d; e, f, g, h\}$.

Repeated multiplication and addition may be indicated by

$$P_1 \cdot \ldots \cdot P_n = \prod_{i=1}^{n} P_i \text{ and } P_1 + \ldots + P_n = \sum_{i=1}^{n} P_i.$$ 

We say that $P_2$ is larger than or equal to $P_1$, written $P_1 \leq P_2$, if and only if every block of $P_1$ is contained of $P_2$.

If $P$ and $P_1$ are partitions on $S$ and $P \geq P_1$, then $P$ defines a partition $P''$ on the set of blocks of $P'$ if we let $B_{P''}(s) = B_{P_1}(t(P''))$ if and only if $s = t(P)$. $P''$ is called the quotient partition of $P$ with respect to $P_1$.

**Example:** Let $P = \{a, b, c, d, e, f, g, h, i\}$ and $P' = \{a, c; b, d, e; f, h; g, i\}$.

Then $P'' = \{B_1, B_2, B_3, B_4\}$ where $B_1 = \{a, c\}, B_2 = \{b, d, e\}, B_3 = \{f, h\}$ and $B_4 = \{g, i\}$.

If $P_1 \geq P'$ and $P_2 \geq P'$, then the quotient partitions with respect to $P'$ satisfy:

1. $P_1 \geq P_2$ if and only if $P_1'' \geq P_2''$.
2. $(P_1 \cdot P_2)'' = P_1'' \cdot P_2''$
3. $(P_1 + P_2)'' = P_1'' + P_2''$

**Partial Ordering and Lattices**

A set of elements $S = \{a, b, c, \ldots\}$ is said to be partially ordered if there exists a relation, indicated as $\leq$, defined on the elements of $S$
which satisfies the following conditions:

1. Reflexive property: \( a \leq a \) for all \( a \) in \( S \)

2. Antisymmetric property: \( a \leq b \) and \( b \leq a \) implies \( a = b \) for all \( a, b \) in \( S \)

3. Transitive property: If \( a \leq b \) and \( b \leq c \) implies \( a \leq c \) for all \( a, b, c \) in \( S \).

If for every \( a, b \) in \( S \) either \( a \leq b \) or \( b \leq a \), we say that \( S \) it totally ordered. If \( S \) is partially ordered set and \( X \) is any subset of \( S \), we say that \( a \) in \( S \) is a lower bound of the set \( X \) if \( a \leq x \) for all \( x \) in \( X \) and that \( a \) is a upper bound of \( X \) is \( a \geq x \) for all \( x \) in \( X \). A lower bound \( b \) of \( X \) is called the greatest lower bound (g.l.b.) of \( X \) if for every \( a \) that is a lower bound of \( X \) we have that \( a \leq b \). Similarly, we define the least upper bound (l.u.b.) of a set \( X \) as the element \( b \) such that \( b \geq x \) for every \( x \) in \( X \) and \( b \leq a \) for all \( a \) that are upper bounds.

If we indicate the l.u.b. and g.l.b. of \( x, y \) in \( S \) as \( x * y \) and \( x + y \), respectively, then the elements of \( S \) must satisfy the following:

1. Idempotent law
   \[ x \cdot x = x \text{ and } x + x = x \]

2. Commutative law
   \[ x \cdot y = y \cdot x \text{ and } x + y = y + x \]

3. Associative law
   \[ x \cdot (y \cdot z) = (x \cdot y) \cdot z \text{ and } x + (y + z) = (x + y) + z \]

4. Absorption law
   \[ x \cdot (x + y) = x \text{ and } x + (x \cdot y) = x \]

**Example:** Let \( S = \{A_0, A_1, A_2, A_3, \emptyset\} \) where \( A_0 = \{a, b, c\} \), \( A_1 = \{a, c\} \), \( A_2 = \{a\} \), \( A_3 = \{c\} \), \( \emptyset = \{\} \).
Figure 1.1 Partial ordering of S

The set $S$, together with the "contained in" relation is also a lattice. We have $A_0 \geq A_1$, $A_1 \geq A_2$, $A_0 \geq A_2$, $A_2 \geq A_0$, $A_2 \geq A_3$, and $A_3 \geq \emptyset$. $A_2$ is not related to $A_3$.

Examining Figure 1.1, we see that any two elements have a l.u.b. and g.l.b., for example:

\[
\begin{align*}
\text{l.u.b.} (A_1, A_3) &= A_1 \\
\text{g.l.b.} (A_1, A_3) &= A_3 \\
\text{l.u.b.} (A_2, A_3) &= A_1 \\
\text{g.l.b.} (A_2, A_3) &= \emptyset 
\end{align*}
\]
CHAPTER II

MACHINES AND STATE ASSIGNMENT PROBLEM

Introduction

The designer of sequential machines with storage (or memory) faces a much more difficult problem than a designer of simple translating machine (machine without storage). Among the many physical devices of sequential machines are digital computers and digital control units.

In this chapter, we will discuss the problem of determining state assignments for finite-state sequential machines. The basic idea is to find methods for selection of assignments in which each binary variable describing the new state depends on a few variables of the old state. The main tool used in this topic is the partition with the substitution property on the set of states of a sequential machine.

Before this is done we shall begin by first defining and introducing a general finite-state sequential machine.

The Machine and State Assignment

A major objective of sequential machine theory is to develop methods for describing and analyzing the dynamic behavior of discrete systems. The behavior of these systems is determined by many constructed forms of storage and combinatorial elements. However, we are not concerned with the internal construction of these (black-boxes) systems. Given any black-box representation of a system if the possible inputs to the system are
sequences of symbols selected from a finite set \( I \), and the resulting outputs are sequences of symbols selected from a finite set \( I \), then we refer to the system as a sequential machine. When theorizing a sequential machine, the input set \( I \), the output set \( O \), and a state (or internal configurations) set \( S \), and the relations between these sets are of fundamental importance. Since the internal construction of the machine is generally not under observation, the state set \( S \) is not easily defined. Hence a more general and mathematically useful definition is in order:

**Definition 2.1:** A sequential machine is quintuple \( M = (S,I,O,g,h) \)

where
1. \( S \) is a finite nonempty set of states;
2. \( I \) is a finite nonempty set of inputs;
3. \( O \) is a finite nonempty set of outputs;
4. \( g: S \times I \rightarrow S \) is called a next-state function;
5. \( h: S \times I \rightarrow O \) is called the output function.

A machine that satisfies the conditions of the foregoing definition is called a mealy machine. A modification of this definition which defines the output mapping \( h \) as restricted to a mapping of \( S \) onto \( O \) is called a Moore machine.

The most common representation of a sequential machine \( M = (S,I,O,g,h) \) is illustrated in Fig. 2.1 where \( S = \{r,s,t\} \), \( I = \{a,b\} \), and \( O = \{0,1\} \). It is called a flow table. The rows correspond to each state, the columns correspond to each input, and the table entries indicate each transition. There is also an output section which includes values for each input.
Let $M$ and $M'$ be sequential machines with $I = I'$. Then $M$ is said to be isomorphic to $M'$ if there exists a one-to-one mapping $f$ of $S$ onto $S'$ such that $g(s, x) = g'[f(s'), x']$ and $f[h(s, x)] = h'[f(s'), x']$ for all $x$ in $I$ and all states $s$ in $S$. For example the machines in Fig. 2.2 are isomorphic under the mapping $f$ which, for each $i$ takes $s_i$ into $s_i'$. Machine isomorphism is the most elementary case of two machines imitating each other through the use of combinatorial circuits.
Definition 2.2: If $M$ and $M'$ are two machines, then the triple $(f, l, q)$ is said to be an assignment of $M$ into $M'$ if and only if $f$ is a mapping of $S$ into nonempty subsets of $S'$, $l$ is a mapping of $I$ into $I'$, $q$ is a mapping from $O'$ into $O$, and these mapping satisfy the following relations:

1. $g'[f(s), l(x)] \subseteq f[g(s, x)]$ for all $s$ in $S$ and $x$ in $I$.
2. $q[h'(s', l(x))] = h(s, x)$ for all $s'$ in $f(s)$ and $x$ in $I$.

Definition 2.3: A machine $M'$ is said to be a realization of machine $M$ if and only if there is an assignment $(f, l, q)$ of $M$ into $M'$. If $M$ and $M'$ are state machines, then we require $f$ and $l$ satisfying condition (1) of Def. 2.2 and such that $f$ maps $S$ into disjoint subsets of $S'$.

We now show two realizations of the machines $A$ of Fig. 2.3.

A more general type $A'$ is illustrated in Fig. 2.4 and the state behavior realization is given in Fig. 2.5.

![Fig. 2.3 Machine A](image1)

![Fig. 2.4 Machine A', A realization of Machine A](image2)
\begin{align*}
  f(1) &= 00 & l(a) &= 0 \\
  f(2) &= 01 & l(b) &= 1 \\
  f(3) &= 10 & q(0) &= 0 \\
  & & q(1) &= 1 \\
  (y_1, y_2) &= x = 0 & x = 1
\end{align*}

\begin{tabular}{|c|c|c|c|c|}
  \hline
  0 & 0 & 1 & 0 & 1 & 0 \\
  0 & 1 & 0 & 0 & 1 & 1 \\
  1 & 0 & 1 & 0 & 0 & 1 \\
  1 & 1 & 0 & 0 & 0 & 1 \\
  \hline
\end{tabular}

Fig. 2.5 Machine A", a state realization of machine A.

Note that the states, inputs, and outputs of machines A' and A" are expressed in terms of binary variables and the functions f, l, and q explain how these are interpreted.

**Definition 2.4:** Input binary variables $X_i$ for $1 \leq i \leq m$, state binary variables $s_j$ for $1 < j < n$, transition functions $Y_j: \{s_1, \ldots, s_n, x_1, \ldots, x_m\} \rightarrow \{0, 1\}$, and output functions $Z_k: \{s_1, \ldots, s_n, x_1, \ldots, x_m\}$ imply $\{0, 1\}$ are said to define the following machine

1. $S = \{(s_1, \ldots, s_n)\}$, the set of all n-tuples on $\{0, 1\}$
2. $I = \{(x_1, \ldots, x_m)\}$
3. $O = \{Z_1, \ldots, Z_k\}$
4. $g(y, x) = Y(y, x)$
5. $h(y, x) = Z(y, x)$.

Let $Y_1$ be the first component in the transition state, $Y_2$ be the second component in the transition state and $Z$ be the output. To express $Y_1$ in terms of the complete product of the $y$'s, we use each complete product which takes on the value of 1. Hence all products corresponding to $Y_1 = 1$ must appear in the expansion of $Y_1$ into the sum of complete products and all other excluded. We conclude that every $Y$ of $n$ variable may be expressed uniquely as a sum of complete products of those $n$ variables:
\[ Y = \sum_{i=0}^{n-1} y_i x_i \] where \( y_i \) is the value of \( Y \) at the combination at which \( x_i \) takes on the value of 1. From the flow table of machine \( A' \) Fig. 2.4:

\[ Y_1 = \overline{y_1} \overline{y_2} x + y_1 \overline{y_2} x + y_1 y_2 x + \overline{y_1} y_2 x \]
\[ = \overline{y_1} \overline{y_2} x + y_1 \overline{y_2} x + (y_1 + \overline{y_1}) y_2 x \]
\[ = \overline{y_1} \overline{y_2} x + y_1 \overline{y_2} x + y_2 x \]
\[ = (\overline{y_1} + y_1) \overline{y_2} x + y_2 x \]
\[ = \overline{y_2} x + y_2 x \]
\[ = (\overline{y_2} + y_2)x \]
\[ = x \]

Also \[ Y_2 = \overline{y_1} \overline{y_2} \overline{x} + y_1 \overline{y_2} \overline{x} + y_1 \overline{y_2} \overline{x} + y_1 y_2 \overline{x} + \overline{y_1} y_2 \overline{x} \]
\[ = \overline{y_1} \overline{y_2} (\overline{x} + x) + y_1 \overline{y_2} (\overline{x} + x) + y_1 y_2 \overline{x} + \overline{y_1} y_2 \overline{x} \]
\[ = \overline{y_1} \overline{y_2} + y_1 \overline{y_2} + y_2 (y_1 \overline{x} + y_1 x) \]
\[ = \overline{y_1} \overline{y_2} + y_2 (y_1 \overline{x} + y_1 x) \]
\[ = \overline{y_2} (y_1 + y_1 \overline{x}) + y_2 (y_1 x + \overline{y_1} x) \]
\[ = \overline{y_2} + y_1 x + \overline{y_1} \overline{x} \]

Hence, machine \( A' \) is defined by the Boolean equations:

\[ Y_1 = x \]
\[ Y_2 = \overline{y_2} + xy_1 + \overline{x} \overline{y_1} \]
\[ Z = y_2 \]

And machine \( A'' \) is defined by the equations:

\[ Y_1 = \overline{y_2} \overline{x} \]
\[ Y_2 = \overline{y_1} x \]
\[ Z = y_1 + y_2 \]

We say that the equations realize a machine \( M \) if they define a machine \( M' \) that realizes \( M \). The previous sets of equations realize machine \( A \). The term "realize" is used because the logical equations
determine a schematic circuit diagram from which an engineer can build a physical device to behave like the machine. The equations for A¹ lead to the diagram of Fig. 2.6. The interpretation of Fig. 2.6 is that, at each signal, the contents of delays \( y₁ \) and \( y₂ \) are released, combined with the input to compute the next values, and these new values are stored in the delays.

In a subtle way, we have come to one of the problems this thesis intends to demonstrate. That is, we want to study the state assignment problem for finite-state sequential machine through the use of binary variables describing the new state depends on as few variables as possible from the old state.

To illustrate the basic ideas, we shall consider two different assignments for the sequential machine A in Fig. 2.7. Recall from Boolean algebra that in \( n \) binary variables there were \( 2^n \) complete products. Since the machine has six internal states, we need three binary variables, \( y₁, y₂, y₃ \) to designate these states. Consider the two binary variables assignments (1) and (2) given in Fig. 2.8. If we substitute these assignments in the flow table given in Fig. 2.7, we obtain the relations shown in Fig. 2.9.
<table>
<thead>
<tr>
<th>Inputs</th>
<th>States</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1</td>
<td>0 3 2 0</td>
<td>0 1</td>
</tr>
<tr>
<td></td>
<td>1 5 2 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2 4 1 0</td>
<td>0 3 0</td>
</tr>
<tr>
<td></td>
<td>3 1 4 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 0 3 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 2 3 0</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.7 Machine A

Fig. 2.8 Assignments (1) and (2) for machine A
For the assignment (1) the binary variables of the new state can be computed from the variables of the old state and the input \( x \) as follows:

- \( Y_1 = \overline{y}_1 \overline{y}_2 y_3 \overline{x} + y_2 \overline{y}_3 \overline{x} + y_2 y_3 x \)
- \( Y_2 = \overline{y}_2 x + \overline{y}_1 \overline{y}_2 y_3 + y_1 y_3 \)
- \( Y_3 = y_1 x + y_2 \overline{y}_3 x + y_2 y_3 \overline{x} + \overline{y}_1 \overline{y}_2 \overline{x} \)
- \( Z = y_2 y_3 \)
For the assignment (2) the relations between the new and old state variables are given by the following equations:

\[ Y_1 = y_1 x + \overline{y}_1 x \]
\[ Y_2 = y_2 \overline{x} \]
\[ Y_3 = \overline{y}_2 \overline{y}_3 \]

It can be seen that \( Y_1 \) does not depend on \( y_2 \) or \( y_3 \) and that \( Y_2, Y_3 \) does not depend on \( y_1 \). These equations are far simpler than the equations relating the new and old state variables for assignment (1). One feels intuitively that the reduction of the number of state variables and inputs on which the state variables depends should simplify the logical circuits in the corresponding realization. Because the structure theory for sequential machines and the general understanding of the logical dependence, this can be viewed as an approach to state assignment problems. Hence the preceding problem provides motivation for our second thesis argument.

**Partitions and State Assignment with Substitution Property.**

We shall study the properties of the assignment (2) in the preceding section in order to be able to select such assignments systematically.

Consider the schematic drawing of assignment (2) Fig. 2.10; there exists two sets, the set consisting of \( Y_1 \) and the set consisting of \( Y_2 \) and \( Y_3 \).
When considering only the variable $Y_1$, we cannot distinguish between the states, 0, 1, and 2, or 3, 4, and 5 (see figures 2.7, 2.3, and 2.9). Since $Y_1$ can be computed if we know old $y_1$ and input $x$, after any number of inputs the $Y_1$ will tell us in which one of the two sets of states the state of $A$ will be contained. Similarly, if we consider only the two last variables $Y_2$ and $Y_3$, we cannot distinguish between the states 0, and 5, or 1 and 4 or 2 and 3, if we know $y_2$ and $y_3$ and know the following input sequence, we can compute the resulting $Y_2$ and $Y_3$. If we know in which one of the three possible sets of the initial state is contained, and if we know the input sequence, we can determine the set which will contain the resulting state.
We can partition the set of states into disjoint subsets in such a way that, if we know in which subset the starting state is contained, we are able to tell in which of these sets of state will be contained after any finite sequence of inputs. (see definition 1.1) Hence, consider the two partitions \( P_1 \) and \( P_2 \) in our example — determined by \( (Y_1) \) and \( (Y_2,Y_3) \) respectively—are \( P_1 = \{0,1,2; 3,4,5\} \) and \( P_2 = \{0,5; 1,4; 2,3\} \). We shall say that the partition \( P \) identifies two elements if and only if these elements are contained in the same block. We shall refer to the partition \( P = 0 \), in which each block consists of a single element, and to the partition \( P = I \) in which all elements are contained in one block as the trivial partitions.

**Definition 2.5:** A partition \( P \) on the set of states of the machine \( M = (S,I,O,g,h) \) is said to have the substitution property if and only if \( s = t(P) \) implies that \( g(s,x) = h(t,a)(P) \) for all \( a \) in \( I \).

We refer to partitions with the substitution property as S.P. partitions. It follows that a partition \( P \) has the substitution property if and only if each input maps blocks of \( P \) into blocks of \( P \).

Since the operation of \( M \) determines unique block-to-block transformations on S.P. partitions \( P \), we can think of these blocks as the states of a new state machine defined by \( P \) and \( M \).

**Definition 2.6:** Let \( P \) be a S.P. partition on the set of states of the machine \( M = (S,I,O,g,h) \). Then the \( P \)-image of \( M \) is the state machine \( M_P = (\{B_p\},I,g_p) \) with \( g_p(B_p,x) = B_p \) if and only if \( g(B_p,x) \subseteq B_p \).

We can think of \( M_P \) as a machine which does only part of the computation performed by \( M \), since it only keeps track of which block of \( P \) contains the state of \( M \).
If \( P \) has S.P. on \( M \) and we know the block of \( P \) which contains the state of \( M \), then we can compute the block of \( P \) to which this state of \( M \) is transformed by any input sequence. \( M \) performs this computation. If a partition \( P \) does not have s.P., then this computation is not possible for some input and sequence and initial block.

To illustrate these ideas, note that the partition \( P_1 = \{0,1,2; \ 3,4,5\} \) and \( P_2 = \{0,5; \ 1,4; \ 2,3\} \) has the substitution property with respect to machine A given in Fig. 2.7 and the schematic drawing Fig. 2.10. If we identify states 0, and 1, then the input "0" leads to states 3 and 5, respectively, and we have to identify these states. Note that the input "1" does not yield any new identifications because the states 0 and 1 both go into the state 2 under this input. Since the pairs of states 0,1, and 3,5 are disjoint, we do not have to add any new pairs because of the transitive property. Repeating the process for the states 3 and 5, we obtain the pairs 1,2, and 4,3. This time we obtain that the states 0,1,2 are identified and 3,4,5 are also identified. Thus \( P_1 \) has S.P. Similarly, we can obtain \( P_2 \).

**Theorem 2.1:** If \( P_1 \) and \( P_2 \) are S.P. partitions on the set of states of a sequential machine \( M \), then so are the partitions \( P_1 \cdot P_2 \) and \( P_1 + P_2 \).

**Proof:** Suppose that \( s = t(P_1) \) and \( s = t(P_2) \). Then by definition 2 \( s \equiv t(P_1 \cdot P_2) \). Also, for any input \( a \) in \( I \), \( g(s,a) = g(t,a)(P_1) \) and \( g(s,a) = g(t,a)(P_2) \) since \( P_1 \) and \( P_2 \) are S.P. But, by definition 2 \( g(s,a) = g(t,a)(P_1 \cdot P_2) \).

To show that \( P_1 + P_2 \) has S.P. recall definition 3. Since \( P_1 \) and \( P_2 \) have S.P., \( g(s,a) = g(s_1,a) = g(s,a)(P_2) \), and since \( P_1 \) and \( P_2 \) are both finer than \( P_1 + P_2 \), we conclude that \( g(s,a) = g(s_1,a)(P_1 + P_2) \).
Similarly \( g(s_1, a) = g(s_2, a)(P_1 + P_2) \)
\( g(s_2, a) = g(s_3, a)(P_1 + P_2) \)
\( \ldots \ldots \ldots \ldots \ldots \)
\( g(s_{n-1}, a) = g(t, a)(P_1 + P_2) \)

Hence, from the transitive property, we have \( g(s, a) = g(t, a)(P_1 + P_2) \).

**Theorem 2.2:** The set of all S.P. partitions on a set of states of a sequential machine \( M \) forms a lattice, under the natural partition ordering.

**Proof:** From theorem 2.1, we have that the set of S.P. partitions on \( S \) of \( M \) is closed under "•" and "+". Thus, the set of all S.P. partitions form a sublattice of the lattice of all partitions on \( S \) and therefore a lattice in the natural ordering of partitions.

**Computation S.P. Partitions**

There are basically two ways of obtaining S.P. partitions. The first way is to work with the machine description until a partition is found which satisfies Definition 2.5. The second way is to combine previously obtained S.P. partitions using the sum and product operations (Definitions 2 and 3). Since the second way is much easier than the first, the most efficient procedures are those which rely most heavily on the second way. Of course, some use must be made of the first way since no nontrivial partitions are known a priori to have S.P. A general procedure follows these two steps:

1. For every pair of states \( s, t \), compute the smallest S.P. partition, \( P_{s,t} \), which identifies the pair.
2. Find all possible sums of the \( P_{s,t} \). These sums constitute all the S.P. partitions.
To illustrate this, consider the sequential machine B Fig. 2.11.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>States</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>x = 1</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

If we identify states 0 and 1, the "0" input leads to 5 and 4 respectively, but "1" input leads to 2 and 3 respectively. Repeat the process for 5 and 4 we are led to 3 and 2. But the "1" input for 5 and 4 yield 6. This implies that 5, 4 and 2, 3 can be associated with 0, 1. Notice that "0" sends 6 to 2 and "1" sends 6 to 4. But, 4 leads to 2 and 6. Therefore we may conclude that the 6 may be blocked alone or associated in the same block as 4. Hence, we have these two S.P. partitions $P_1 = \{0, 1; 2, 3; 4, 5, 6\}$, $P_2 = \{0, 1; 2, 3; 4, 5, 6\}$. $P_1 + P_2 = P_2$. Therefore, $P_1$ and $P_2$ are all the partitions with S.P. property.
Further Explanation Through Selected Problems

Selected Problem I

Definition 3.1: Two partitions \( P, P' \), defined on the states of a machine \( S \), are called a partition pair if for each block \( B_i \) of \( P \) and for all input \( x \) in \( I \), \( g(x, B_i) = B'_i \), a block of \( P' \), depends only on \( x \) and \( B_i \).

The composite machine \( M_c \) shown in Fig. 2.12 has the following state assignment and transition table.

<table>
<thead>
<tr>
<th>States ( M_c )</th>
<th>State ( M )</th>
<th>Transition Table of ( M_c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_1 \ y_2 \ y_3 )</td>
<td>( y_1 )</td>
<td>( y ) 0 1 2</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>1 2 7 0</td>
</tr>
<tr>
<td>1 1 2</td>
<td>2</td>
<td>1 2 7 1</td>
</tr>
<tr>
<td>1 2 1</td>
<td>3</td>
<td>1 2 7 1</td>
</tr>
<tr>
<td>1 2 2</td>
<td>4</td>
<td>1 2 7 0</td>
</tr>
<tr>
<td>2 1 1</td>
<td>5</td>
<td>1 2 7 0</td>
</tr>
<tr>
<td>2 1 2</td>
<td>6</td>
<td>1 2 7 0</td>
</tr>
<tr>
<td>2 2 1</td>
<td>7</td>
<td>1 2 7 1</td>
</tr>
<tr>
<td>2 2 2</td>
<td>8</td>
<td>1 2 7 0</td>
</tr>
</tbody>
</table>

Fig. 2.12 State Assignment and transition table.

Fig. 2.13 Machine \( M_c \)
(a) Find all S.P. partitions and partition pairs associated with the reduced state table of $M_c$ which can be obtained by direct inspection of Fig. 2.12.

(b) Find the transition table for $M_2$ submachine.

Solution: (a) S.P. partitions

$$P_{y_1} = \{\bar{y}_1, \bar{y}_2, \bar{y}_3, \bar{y}_4; \ \bar{y}_5, \bar{y}_6, \bar{y}_7, \bar{y}_8\}$$

$$P_{(y_1, y_2)} = \{\bar{y}_1, \bar{y}_2; \ \bar{y}_3, \bar{y}_4; \ \bar{y}_5, \bar{y}_6; \ \bar{y}_7, \bar{y}_8\}$$

$P$ = Trivial partition

Partition pairs: $(P_{y_1}, P_{y_2})(P_0, P_y)$

(b) $M_2$ Table

<table>
<thead>
<tr>
<th>$y_2$</th>
<th>$(0,1)$</th>
<th>$(0,2)$</th>
<th>$(1,1)$</th>
<th>$(1,2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 2.14
Selected Problem 2:

Find all S.P. partitions for machine M of Fig. 2.15, and plot the lattice.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 2.15 Machine M

Solution: $P_0 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_2 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_3 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_4 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_5 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$P_6 = \{1, 2, 3, 4, 5, 6, 7, 8\}$
$I = \{1, 2, 3, 4, 5, 6, 7, 8\}$
Selected Problem 3: Let \((P_1, P_2)\) and \((N_1, N_2)\) be partition pairs. Show that \((P_1 \cdot N_1, P_2 \cdot N_2)\) and \((P_1 + N_1, P_2 + N_2)\) are also partition pairs.

Solution: If \(s = t(P_1 \cdot N_1)\), then \(s = t(P_1)\) and \(s = t(N_1)\). Therefore for any input \(x\), \(g(s, x) = g(t, x)(P_2)\) and \(g(s, x) = g(t, x)(N_2)\). Hence \(g(s, x) = g(t, x)(P_2 \cdot N_2)\), which show that \((P_1 \cdot N_1, P_2 \cdot N_2)\) is a partition pair.

If \(s \equiv t(P_1 + N)\) then there exists a sequence of states \(s = s_0, s_1, \ldots, s_n = t\) such that \(s_i \equiv s_{i+1}(P_1)\) for \(i\) even and \(s_i \equiv s_{i+1}(N)\) for \(i\) odd. Therefore, for any input \(x\), \(g(s_i, x) = g(s_{i+1}, x)(P_2)\) for \(i\) even and \(g(s_i, x) = g(s_{i+1}, x)(N_2)\) for \(i\) odd, hence, \(g(s, x) = g(t, x)(N_2)\) by transitivity and we conclude that \((P_1 + N_1, P_2 + N_2)\) is a partition pair.
CHAPTER III

PAIR ALGEBRA AND DECOMPOSITION

Pair Algebra

We should like to establish a basic mathematical framework that may be used when slight variations of pair algebra are given.

Definition 3.1: Let \( L_1 \) and \( L_2 \) be finite lattices. Then a subset \( D \) of \( L_1 \times L_2 \) is a pair algebra on \( L_1 \times L_2 \) if and only if the two following conditions hold:

1. \((x_1, y_1) \) and \((x_2, y_2) \) in \( D \) implies that \((x_1 \cdot x_2, y_1 \cdot y_2) \) and \((x_1 + x_2, y_1 + y_2) \) are in \( D \).
2. For any \( x \) in \( L_1 \) and \( y \) in \( L_2 \), \((x, 0) \) and \((0, y) \) are in \( D \), where the lattice operations of g.l.b. and l.u.b. are denoted by "\( \cdot \)" and "\( + \)" respectively, and the zero and identity elements are denoted by "\( 0 \)" and "\( I \)".

The set of partition pairs discussed in the previous chapter forms a pair algebra, where the lattices \( L_1 \) and \( L_2 \) are both identical to the partition lattice on the set of states of a machine (see selected problem 1).

Theorem 3.1: If \( D \) is a pair algebra on \( L_1 \times L_2 \) and \((x, y) \) is in \( D \), then \( x' \leq x \) and \( y \leq y' \) implies that \((x', y), (x, y'), \) and \((x', y') \) are in \( D \).

Proof: By (2) of definition 3.1 \((x', I) \) is in \( D \). By (1) of definition 3.1 \((x', I) \) and \((x, I) \) in \( D \) implies that \((x' \cdot x, 1 \cdot y) = (x', y) \) is in \( D \). The other two cases follow by a similar argument.

Hence, if a pair is in the pair algebra \( D \) on \( L_1 \times L_2 \), we can obtain another pair by replacing the first component by a smaller element, and for the second component by a larger element. The next definitions will
characterize the largest possible first component of a pair in D and the smallest possible second component.

**Definition 3.2:** Let D be a pair algebra on $L_1 \times L_2$. For $x$ in $L_1$ we define $m(x) = \bigcap \{ y_1 : (x, y_1) \in D \}$. For $y$ in $L_2$ we define $M(y) = \bigcup \{ x_1 : (x_1, y) \in D \}$.

**Definition 3.3:** For $(x, y)$ and $(x', y')$ in $L_1 \times L_2$ we define $(x, y) \leq (x', y')$ if and only if $x \leq x'$ in $L_1$ and $y \leq y'$ in $L_2$.

**Definition 3.4:** An element $(x, y)$ in a pair algebra D is called an Mm pair if and only if $y = m(x)$ and $x = M(y)$.

**Theorem 3.2:** If D is a pair algebra, then:

1. $[M(y), y]$ and $[x, m(x)]$ are in D
2. $x_1 \geq x_2$ implies that $m(x_1) \geq m(x_2)$
3. $m(x_1 + x_2) = m(x_1) + m(x_2)$
4. $m(x_1 \cdot x_2) \leq m(x_1) \cdot m(x_2)$
5. $y \geq m(x)$ if and only if $(x, y)$ in D
6. $y_1 \geq y_2$ implies that $M(y_1) \geq M(y_2)$
7. $M(y_1 + y_2) \geq M(y_1) + M(y_2)$
8. $M(y_1 \cdot y_2) = M(y_1) \cdot M(y_2)$
9. $x \leq M(y)$ if and only if $(x, y)$ in D
10. $M(m(x)) \geq x$
11. $m(M(y)) \leq y$
12. $\big\{ M(y), m(y) \big\} = M(y)$
13. $\big\{ M(x), m(x) \big\} = m(x)$
14. $\big\{ m(M(y)), m(y) \big\} = M(y)$ and $\{ M(m(x)), m(x) \}$ are Mm pairs in D
15. if $(x_1, y_1)$ and $(x_2, y_2)$ are Mm pairs of D, then $x_1 \leq x_2$ if and only if $y_1 \leq y_2$
The set of $Mm$ pairs of $D$ forms a lattice in which
\[ g.l.b. \left\{ (x_1, y_1), (x_2, y_2) \right\} = [x_1 \cdot x_2, m(x_1 \cdot x_2)], \]
\[ l.u.b. \left\{ (x_1, y_1), (x_2, y_2) \right\} = [M(y_1 + y_2), (y_1 + y_2)]. \]

\textbf{Corollary 3.1:} If $D$ is a pair algebra on $L \times L$, then $L = \{ x \in L : m(x) \leq x \leq M(x) \}$ is a sublattice of $L$.

If $D$ is a pair algebra of all partition pairs on a sequential machine $M$, the lattice $L$ consists of all the partitions with the substitution property which are related to homomorphisms and play an important role in decomposition theory of sequential machines.

If $D$ is defined on $L \times L$ let $m'(x) = m(x)$ and $m^k(x) = m[m^{k-1}(x)]$ for $k > 1$. Similarly define powers of $M(y)$.

\textbf{Corollary 3.2:} If $D$ is a pair algebra on $L \times L$, then $m^k(x) = 0$ if and only if $x \leq M^k(0)$.

\textbf{Proof:} If $m^k(x) = 0$, then $M^k(0) = M^k[m^k(x)] \geq M^{k-1}(m^{k-1}(x)) \geq x$, using the relation (Th. 3.2 (10)) $M[m(x)] \geq x$. Conversely $x \leq M^k(P)$ implies by a similar argument (using Th. 3.2 (11)) that $m^k(x) \leq m^k[M^k(0)] \leq 0$.

\textbf{Decomposition}

In this section we shall investigate some of the applications of pair algebra to sequential machines. Recall the definition of a sequential machine (Def. 2.1). We shall now define a submachine.

\textbf{Definition 3.5:} A machine $M' = (S', I', 0', g', h')$, is a submachine of the machine $M = (S, I, 0, g, h)$ if and only if restricted $S' \subseteq S$, $I' \subseteq I$, $0' \subseteq 0$

\[ g' = g \text{ restricted to } I' \times S \text{ and } h' = h \text{ restricted to } I' \times S'. \]
Definition 3.6: A machine $M' = (S', I', O', g', h')$ is said to be a homomorphic image of the machine $M = (S, I, O, g, h)$, if and only if there exist three many one mappings:

\[ H_1: S \rightarrow S', \]
\[ H_2: I \rightarrow I', \]
\[ H_3: O \rightarrow O'. \]

such that $H_1[g(a, s)] = g'[H_2(a), H_1(s)]$ and $H_3[h(a, s)] = g'[H_2(a), H_1(s)]$.

Definition 3.7: We shall say that $M_1$ realizes $M_2$ if and only if the reduced form of $M_2$ is homomorphic image of a submachine of $M_1$.

Definition 3.8: The serial connection of two machines $M_1 = (S_1, I_1, O_1, g_1, h_1)$ and $M_2 = (S_2, I_2, O_2, g_2, h_2)$ for which $O_1 = I_2$ is the machine $M = (S_1 \times S_2, I_1 \times I_2, O_1 \times O_2, g, h)$, where $g[(a, b), (s, t)] = \{g_1(a, s), g_2[h_1(a, s), t]\}$ and $h[(a, b), (s, t)] = [h_1(a, s), h_2(b, t)]$.

Theorem 3.3: The set of all partition pairs on $M = (S, I, O, g, h)$ forms a pair algebra $D'$ on $L \times L$ where $L$ is the lattice of all partitions on the set $S$.

Corollary 3.3: The set of partitions on $S$ of $M,M = \{P: m(P) \leq P\}$ forms a sublattice of the lattice of all partitions on $S$.

Note that such partitions were defined on sequential machines in the previous chapter and are referred to as S.P. partitions. The next two theorems show that S.P. partitions play a dominant role in the study of machine decomposition.

Theorem 3.4: The state behavior of $M$ can be realized by a serial connection of two smaller machines $M_1$ and $M_2$ if and only if there exists a nontrivial S.P. partition $P$ on $S$ of $M$. 
Theorem 3.5: The state behavior of \( M \) can be realized by a parallel connection of two smaller machines \( M_1 \) and \( M_2 \) if and only if there exists two nontrivial S.P. partitions, \( P_1 \) and \( P_2 \) on \( S \) of \( M \) such that \( P_1 \cdot P_2 = 0 \).

We shall now apply the principles developed in this and the previous section to an analysis of machine \( D \) given in Fig. 3.1.

\[
\begin{array}{c|cccccc}
\text{x_1x_2 inputs} & 00 & 01 & 11 & 10 & 1 & Z \\
\hline
1 & 3 & 1 & 4 & 2 & 0 \\
2 & 1 & 5 & 4 & 2 & 0 \\
3 & 3 & 4 & 3 & 5 & 0 \\
4 & 5 & 1 & 4 & 2 & 0 \\
5 & 5 & 4 & 3 & 5 & 1 \\
\end{array}
\]

Fig. 3.1 Machine D

First we calculate the \( m(P_{12}) \). To find \( m(P_{12}) \) we look at the flow table and see that to identify rows 1 and 2, we must set states 1 and 3, 1 and 5 equivalent and by transitivity, then we must also have 3 and 5 equivalent. Thus \( m(P_{12}) = \{1, 3, 5; \overline{2}; \overline{4}\} \). Similarly, we arrive at the following list of \( m(P_{ab}) \) for \( D \):

\[
m(P_{12}) = \{1, 3, 5, \overline{2}, \overline{4}\} = P_1 \,'
\]

\[
m(P_{13}) = m(P_{45}) = \{1, 3, 4; \overline{2}, \overline{5}\} = P_2 \,'
\]

\[
m(P_{14}) = m(P_{35}) = \{1; \overline{2}; \overline{3}, 5; \overline{4}\} = P_3 \,'
\]

\[
m(P_{15}) = m(P_{23}) = m(P_{25}) = m(P_{34}) = 1
\]

\[
m(P_{24}) = \{1, 5; \overline{2}; \overline{3}; \overline{4}\} = P_4 \,'
\]

we may forget about the table and work with these \( m(P_{ab}) \).
To compute the list of \( m \)-partitions for machine D we consider all possible sums of the \( m(P_{ab}) \). Adding non-empty subsets of them, we do not get any additional partitions. Summing over the empty set, we get the zero partition. This completes our list of \( m \)-partitions now we start work on the \( M \)-partitions: 

\[
M(P_{1'}) = P_{ab} = P_{12}P_{14} + P_{35} + P_{24}
\]

summed over \( m(P_{ab}) \leq P_{1''} \). Thus we get all the \( M \)-partitions:

\[
M(I) = I
\]

\[
M(P_1') = P_1 = \left\{ \overline{1,2,4}; \overline{3,5} \right\}
\]

\[
M(P_2') = P_2 = \left\{ \overline{1,3}; \overline{2}; \overline{4,5} \right\}
\]

\[
M(P_3') = P_3 = \left\{ \overline{1,4}; \overline{3,5}; \overline{2} \right\}
\]

\[
M(P_4') = P_4 = \left\{ \overline{1}; \overline{2,4}; \overline{3}; \overline{5} \right\}
\]

\[
M(0) = 0
\]

Combining the previous results we obtain a complete list of the \( Mm \) pairs and these contain the information about the information flow in the machine.

The \( Mm \) lattice is given in Fig. 3.2 and the \( Mm \) pairs given below:

\[
(P_1, P_1') = (\left\{ \overline{1,2,4}; \overline{3,5} \right\}, \left\{ \overline{1,3,5}; \overline{2}; \overline{4} \right\})
\]

\[
(P_2, P_2') = (\left\{ \overline{1,3}; \overline{2}; \overline{4,5} \right\}, \left\{ \overline{1,3,4}; \overline{2,5} \right\})
\]

\[
(P_3, P_3') = (\left\{ \overline{1,4}; \overline{3,5}; \overline{2} \right\}, \left\{ \overline{1}; \overline{2}; \overline{3,5}; \overline{4} \right\})
\]

\[
(P_4, P_4') = (\left\{ \overline{1}; \overline{2,4}; \overline{3}; \overline{5} \right\}, \left\{ \overline{1,5}; \overline{2}; \overline{3}; \overline{4} \right\})
\]

\[
(0,0)
\]

\[
(I, I)
\]
Fig. 3.2 MM lattice for machine D.

It is interesting to observe that the partitions with the S.P. are easily obtained from this list. If \( P \) has S.P., then \( M(P) \geq P \geq m(P) \).

In this case, \( P \) can satisfy \( P_3 \geq P \geq P_3' \), \( I \geq P \geq l \), or \( 0 \geq P \geq 0 \). Hence there are two nontrivial partitions with S.P., they are 1; 4; 2; 3, 5 and 1; 2; 3, 5; 4.

The partition algebra is a very flexible concept and is actually more fundamental than the partition pair concept with which it was first introduced. There are several other problems which can be stated and solved in terms of some appropriate pair algebra, but there is little to be gained by enumerating them. The point is that pair algebra and its application to sequential machines have great potential toward solving larger problems.
BIBLIOGRAPHY

Books


Articles


