010261-6-T

Augmentation of Machine Structure to Improve Its Diagnosability

L. HSIEH

under the direction of
Professor J. F. Meyer

July 1973

Prepared under
NASA Grant NGR23-005-463

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SYSTEMS ENGINEERING LABORATORY

THE UNIVERSITY OF MICHIGAN, ANN ARBOR

THE UNIVERSITY OF MICHIGAN

SYSTEMS ENGINEERING LABORATORY

Department of Electrical and Computer Engineering
College of Engineering

SEL Technical Report No. 71

AUGMENTATION OF MACHINE STRUCTURE TO IMPROVE ITS DIAGNOSABILITY

by

Lo Hsieh

Under the direction of
Professor John F. Meyer

July 1973

Prepared under

NASA Grant
NGR23-005-463

CONTENTS

I. INTRODUCTION AND FUNDAMENTALS

With the rapid advances of modern electronic technology, thousands of logic elements can now be incorporated into a single large scale integrated (LSI) circuit chip. In order to ensure that a LSI chip functions correctly according to its design, it is necessary to test the chip after it has been manufactured and, periodically, after it is in use. Since the number of input and output pins (and hence that of the test terminals) on each LSI chip are relatively small and, furthermore, the behavior of the circuits on most such chips are sequential in nature, the diagnostic problem becomes a difficult one. Fault diagnosis of sequential machines has drawn growing attention lately [1]-[9]. It has been shown that checking sequences are among the most general diagnostic sequences for sequential machines under relatively unrestricted faults. It has also been pointed out [4] that sequential machines, which have repeated symbol distinguishing sequences (RDS's), have short checking sequences. This report is concerned with the further study of these two types of diagnosabilities, that is, i) the possession of a checking sequence (i.e., "checkability") and ii) the possession of an RDS. In particular, the investigation is concerned with methods of augmenting the structure of a sequential machine such that the augmentation can realize the behavior of the given machine and, in addition, is diagnosable in the sense of i) or ii).

To precisely formulate the concept of a diagnostic test when faults are unrestricted, we adopt the following notation and terminology in describing the structure and behavior of sequential machines. $M = (I, Q, Z, \delta, \lambda)$ will denote a (Mealy) sequential machine with finite input alphabet I, state set Q, finite output alphabet Z, transition function $\delta: Q \times I \longrightarrow Q$ and output function: $\lambda: Q \times I \longrightarrow Z$. (M is <u>finite-state</u> if Q is finite.) If X is a set, X^+ will denote the set of all finite, non-null sequences over X; the set $X^* = X^+ \cup \{\Lambda\}$ includes the null sequence Λ. If $\bar{\delta}$ and $\bar{\lambda}$ are the natural extensions of δ to $Q \times I^*$ and λ to $Q \times I^+$, the <u>state behavior of M in q</u> ($q \in Q$) is the function $\alpha_q: I^* \longrightarrow Q$ where $\alpha_q(x) = \bar{\delta}(q, x)$, for all $x \in I^+$, and $\alpha_q(\Lambda) = q$; the (input-output) <u>behavior of M in q</u> is the function $\beta_q: I^+ \longrightarrow Z$ where $\beta_q(x) = \bar{\lambda}(q, x)$, for all $x \in I^+$. (Thus $\beta_q(xa) = \lambda(\alpha_q(x), a)$, for all $x \in I^*$, $a \in I$). The <u>behavior of M</u> is the set $B_M = \{\beta_q \mid q \in Q\}$. If $q, r \in Q$, r is <u>reachable from q</u> if there is an input sequence $x \in I^*$ such that $\alpha_q(x) = r$, and $R(q)$ will be used to denote the set of all states in Q that are reachable from q. M is <u>reachable from q</u> if r is reachable from q, for all $r \in Q$. M is <u>strongly connected</u> if M is reachable from q, for all $q \in Q$. Regarding input-output behavior, if $q, r \in Q$, q is <u>equivalent</u> to r ($q \equiv r$) if $\beta_q = \beta_r$ (i.e., $\beta_q(x) = \beta_r(x)$, for all $x \in I^+$). M is <u>reduced</u> if $q \equiv r$ implies $q = r$, for all $q, r \in Q$. If $q \in Q$, we will sometimes refer to the pair (M, q) as an <u>initial state machine</u> with initial state q, and refer to β_q as the <u>initial state (input-</u>

output) behavior of (M,q). (M,q) is reachable if M is reachable from q, and (M,q) is reduced if M is reduced. The reachable submachine of M from $r \in Q$, denoted $(\overline{M, r})$, is the machine $(I, R(r), Z, \delta|R(r) \times I, \lambda|R(r) \times I)$, where $f|X$ means the function f restricted to the set X. If M and M' are sequential machines over the same input alphabet I and output alphabet Z, $q \in Q$, and $q' \in Q'$ then q is equivalent to $q'(q \equiv q')$ if $\beta_q = \beta'_{q'}$; q and q' are distinguishable if they are not equivalent; M is equivalent to $M'(M \equiv M')$ if $B_M = B'_{M'}$.

Given a machine M, a state $q \in Q$, and an input sequence $x \in X^+$, the value $\beta_q(x)$ of the behavior of M in q is interpreted as the last ouput symbol emitted if input sequence x is applied, starting in state q. In discussing the diagnosis of machine, it is often convenient to have an explicit representation of a whole output sequence as opposed to the last symbol. Accordingly, the extended (or sequence-to-sequence) behavior of M in q is the function $\hat{\beta}_q : I^+ \longrightarrow Z^+$ where $\hat{\beta}_q(a_1 a_2 \ldots a_n) = \beta_q(a_1)\beta_q(a_1 a_2)\ldots \beta_q(a_1 a_2 \ldots a_n)$. Note that, since M is a Mealy machine, the length of the output sequence $\hat{\beta}_q(x)$ is equal to the length of x. Since β_q uniquely determines $\hat{\beta}_q$, one should also note that, as functions, $\beta_q = \beta_r$ if and only if $\hat{\beta}_q = \hat{\beta}_r$.

For each sequential machine $M = (I, Q, Z, \delta, \lambda)$ we associate with it a (state) graph whose nodes are the states in Q, having an arc from state q to state r labeled with a/b, where $a \in I$ and $b \in Z$, if and only if $\delta(q, a) = r$ and $\lambda(q, a) = b$. We will use $M|a$ to denote the autonomous machine $(\{a\}, Q, Z, \delta|Q \times \{a\}, \lambda|Q \times \{a\})$, and D_a to denote its state graph.

In the state graph of an autonomous machine, the input label on each arc will be omitted.

Let D be a graph of a sequential machine M. A subgraph of D consists of nodes and arcs of D. A path in D is a sequence of arcs $q_1q_2, q_2q_3, \ldots, q_{k-1}q_k$, such that all the nodes, q_i's, are distinct; the length of this path is k-1 which is the number of arcs in this path. A cycle in D is a closed path, i.e. it is a path with identical first and last node. The period of a cycle is the number of arcs in the cycle. A semipath in D is a sequence of arcs $\overline{q_1q_2}, \overline{q_2q_3}, \ldots, \overline{q_{k-1}q_k}$, where $\overline{q_iq_{i+1}}$ means an arc q_iq_{i+1} or $q_{i+1}q_i$, such that all q_i's are distinct. A semicycle in D is a closed semipath. A graph (or subgraph) is strongly connected if there is a path from every node to every other node, and it is weakly connected if between any two nodes there is a semipath. A graph is disconnected if it is not weakly connected.

Let us now consider the diagnosis of a machine $M = (I, Q, Z, \delta, \lambda)$ relative to the class of machines

$$\mathfrak{M}(M) = \{M' \mid M' = (I, Q', Z, \delta', \lambda') \text{ and } |Q'| \leq |Q|\}$$

(where $|X|$ denotes the cardinality of set X).

Thus, with respect to this class of faulty machines, a fault of M is any abnormality which can result in any machine M' with the same input and output alphabets as M and no more states than M. The choice of $\mathfrak{M}(M)$ is motivated by certain physical assumptions; namely that the faulty system is representable by a machine and has the same input and output terminals as the fault-free system. Q is chosen to represent all possible physical states of the fault-free system; hence no permanent fault will cause an increase in the number of physical states. (It is possible, however, that the number of nonequivalent states <u>will</u> increase since M need not be reduced.) Under these assumptions, faults are unrestricted in the sense that any permanent fault of the system represented by M results in a system representable by a machine $M' \in \mathfrak{M}(M)$. The type of diagnostic tests for machines with unrestricted faults are "checking sequences" and "detecting sequences." These sequences are defined formally below.

<u>Definition 1.1</u>

If M is a machine, $q \in Q$ and $x \in I^+$, then x is a <u>checking sequence (CS) for (M, q)</u> if, for all $M' \in \mathfrak{M}(M)$ and all $q' \in Q'$, $\hat{\beta}'_{q'}(x) = \hat{\beta}_q(x)$ implies that, for some state $r' \in Q'$, $r' \equiv q$ (i.e., $\beta'_{r'} = \beta_q$).

<u>Definition 1.2</u>

If $M = (I, Q, Z, \delta, \lambda)$ is a machine, $q \in Q$, and $x \in I^+$, then x is a <u>detecting sequence for (M, q)</u> if, for all $M' \in \mathfrak{M}(M)$ and all $q' \in Q'$, $\hat{\beta}'_{q'}(x) = \hat{\beta}_q(x)$ implies $q' \equiv q$ (i.e., $\beta'_{q'} = \beta_q$).

It is easily seen that detecting sequences are special cases of checking sequences. The difference between a checking sequence and a detecting sequence is that in the latter case, a positive response to the sequences says that the state of M' just before application of the sequence <u>is</u> a state equivalent to q; in the former case we can only guarantee that some state of M is equivalent to q. For more detailed study on properties of checking sequences and detecting sequences of sequential machines, the reader is referred to [8]. A sequential machine is checkable (detectable) if it has a checking (detecting) sequence.

Since the checkability or detectability of a machine (M, q) depends on the behavior β_q let us focus our attention, for the moment, on behavior, per se. Suppose that $\beta: I^+ \longrightarrow Z$ and let us consider all other functions $\beta': I^+ \longrightarrow Z$ that can be derived from β by first applying a fixed sequence $y \in I^*$. More precisely,

Definition 1.3

If $\beta: I^+ \longrightarrow Z$ and $y \in I^*$ the derivative of β with respect to y is the function $\beta_y: I^+ \longrightarrow Z$ where $\beta_y(x) = \beta(yx)$, for all $x \in I^+$.

The concept of a derivative (although it may go by some other name or even remain nameless) is essentially the concept of "state" from a behavioral point of view (see [12], for example). The name is borrowed from Brzozowski [13] since his use of the term, if translated from sets to functions, is a special case of the above definition. A fundamental relationship, that provides a machine interpretation of derivatives, follows directly from the above definition and the definition of machine behavior, that is:

If β_q is the behavior of M in q, then $(\beta_q)_x$ is the behavior of M in $\alpha_q(x)$.

In what follows, we will let D_β denote the set of all derivatives of β, that is

$$D_\beta = \{\beta_y \mid y \in I^*\}.$$

Given an arbitrary $\beta\colon I^+ \longrightarrow Z$, the derivatives of β permit an easily conceived construction of a "canonical" sequential machine that realizes β, where the states of the machine are just the derivatives D_β. More precisely, if $\beta\colon I^+ \longrightarrow Z$, the <u>derivative machine of β</u> is the sequential machine

$$M_\beta = (I, Q_\beta, Z, \delta_\beta, \lambda_\beta)$$

where $Q_\beta = D_\beta$ and for all $\beta_y \in Q_\beta$ and all $a \in I$,

$$\delta_\beta(\beta_y, a) = \beta_{ya},$$

$$\lambda_\beta(\beta_y, a) = \beta(ya).$$

The reader is referred to [12] for details as to how the construction is motivated and for verification of the important properties summarized in the following theorem:

<u>Theorem 1.1</u>

If $\beta\colon I^+ \longrightarrow Z$ then M_β is reduced, reachable from β, and for all $\beta' \in D_\beta$, the behavior of M_β in β' is β'.

Given a machine M. Let $q \in Q$. We will see that the state set of M_{β_q} (the derivative machine of β_q), can be expressed in another way. Let $M' = (I, Q', Z, \delta', \lambda')$, where

$$Q' = \{ \beta_r \mid r \in R(q), \text{ i.e. the reachable part of M from } q\},$$

and δ' and λ' are defined as, $\forall\, \beta_r \in Q'$ and $a \in I$,

$$\delta'(\beta_r, a) = \beta_{\alpha_r(a)}$$

and

$$\lambda'(\beta_r, a) = \beta_r(a).$$

Since, by definition, $\forall\, \beta_r \in Q'$, $\exists\, x \in I^*$ such that $\alpha_q(x) = r$, the above definitions for δ' and λ' can be rewritten as, $\forall\, x \in I^*$ and $a \in I$,

$$\delta'(\beta_{\alpha_q(x)}, a) = \beta_{\alpha_q(xa)}$$

and

$$\lambda'(\beta_{\alpha_q(x)}, a) = \beta_q(xa).$$

Since, by definitions of β_q and $(\beta_q)_x$, $\forall\, x \in I^*$ and $y \in I^+$,

$$\beta_{\alpha_q(x)}(y) = \beta_q(xy)$$

and

$$(\beta_q)_x(y) = \beta_q(xy),$$

we conclude that

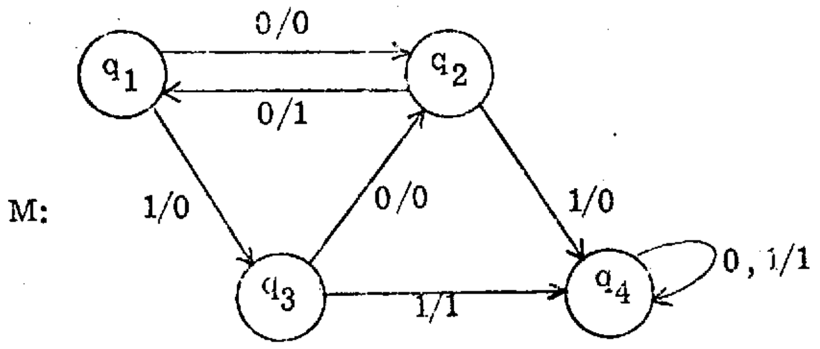
$$\beta_{\alpha_q(x)} = (\beta_q)_x \,.$$

Clearly $\beta_{\alpha_q(x)}$ and $(\beta_q)_x$ are states of Q' and Q_{β_q}. We can easily see that $\delta' = \delta_{\beta_q}$ and $\lambda' = \lambda_{\beta_q}$. Therefore M_{β_q} and M' are actually the same machine. In our following studies, we will use either $\{(\beta_q)_x \mid x \in I^*\}$

or $\{\beta_r \mid r \in R(q)\}$, whichever is more convenient, to denote the state set of M_{β_q}.

To define an important machine structural properties, namely simple connectivity, we need a few more notions and terminologies. A subset R of the state set Q of a machine M is strongly connected if for any $r_1, r_2 \in R$, r_2 is reachable from r_1. A strongly connected subset R of Q is maximal if no other strongly connected subset of Q includes R as a subset. A component C of M is a maximal and strongly connected subset of Q.

For example, for machine M, $\{q_1, q_2\}$ is



strongly connected but not maximal because $\{q_1, q_2\}$ is included in $\{q_1, q_2, q_3\}$ which is maximal and strongly connected.

Definition 1.4

A machine (M, q_1) is simply connected if it has a state diagram consisting of unidirectionally and linearly connected components as shown in Figure 1.1, where C_i's are components of M, each arc represents a single input induced transition, and $\bigcup_{1 \leq i \leq k} C_i = Q$. The states q_i and r_i are called respectively the <u>initial state</u> and the <u>end state of C_i</u>.

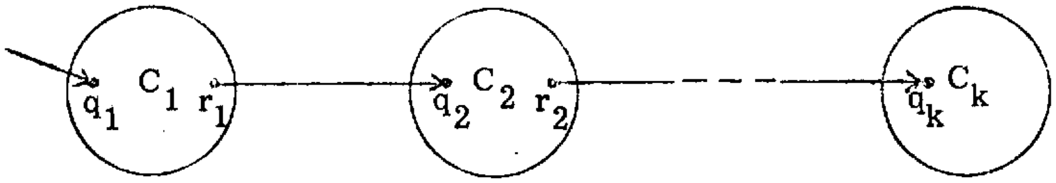


Figure 1.1 A Simply Connected Machine

Another characterization of simple connectedness is given in the following theorem. Its proof can be found in [8].

<u>Theorem 1.2</u>

A machine (M, q) is simply connected if and only if any binary partition of Q into two sets S_1 and S_2, where $q \in S_1$, satisfies one of the following conditions:

(1) There are transitions going both way between S_1 and S_2.

(2) There is only one transition from S_1 to S_2.

II. CHECKABLE REALIZATIONS OF MACHINES

Section 2.1 Introduction

As our previous investigations revealed [8], many systems are intrinsically better suited for fault-diagnosis than others. It should be useful if one knows how to design machines that, besides being able to perform certain tasks, also have better fault-diagnosabilities than other machines which can do the same tasks. This chapter concerns the question of how to improve one aspect of fault-diagnosabilities, namely the checkability, of a machine. Specifically, the question that we like to study is: Given a noncheckable machine, can we design a checkable machine which is capable of simulating the input-output behavior of the given machine in real-time?

Intuitively, if a machine M is structurally minimal, then any machine, if it exists, that can mimic the behavior of M and has an additional property of being checkable, should have a larger structure than that of M. This intuition will be shown to be true. Various methods of augmenting the machine structure will be examined and the answer to the above question by each of these methods will be presented.

We begin with the introduction of a formal definition for input-output simulation in real time of a machine by another machine. It is called realization by Meyer and Zeigler [14]. Let $M' = (I', Q', Z', \delta', \lambda')$ and $M = (I, Q, Z, \delta, \lambda)$ be two sequential machines. Then:

Definition 2.1

M' realizes M if there is a triple of functions $(\sigma_1, \sigma_2, \sigma_3)$ where $\sigma_1: I^+ \longrightarrow I'^+$ such that $\sigma_1(I) \subseteq I'^+$, $\sigma_2: Q \longrightarrow Q'$, $\sigma_3: Z'' \longrightarrow Z$ where $Z'' \subseteq Z'$, such that, for all $q \in Q$ and all $x \in I^+$, $\beta_q(x) = \sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(x)))$. M' is said to be a realization of M if M' realizes M. σ_1 and σ_3 are called the input encoding and output decoding functions.

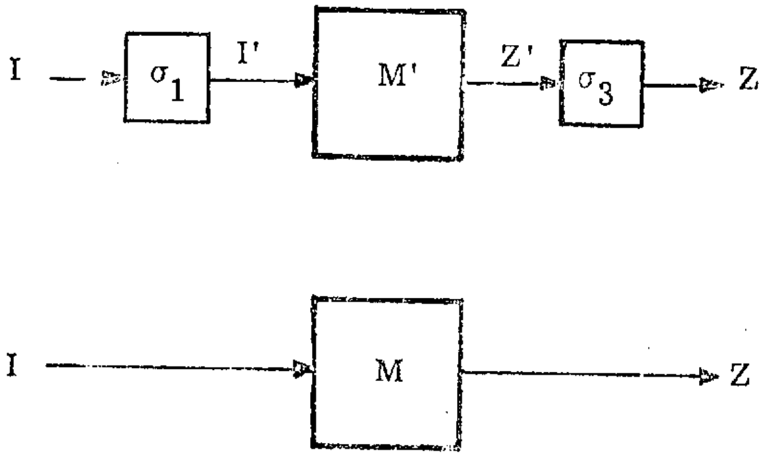


Figure 2.1 M' Realizes M

It has been shown by Leake [15] that the above behavioral definition is equivalent to the more structurally oriented definition of Hartmanis and Stearns [16] in terms of an "assignment" of M into M'.

A few easily verified properties of machine realizations are included in the following theorem without giving the proof.

Theorem 2.1

A sequential machine M realizes its reduced form M_R and vice versa. If M_1 realizes M_2 and M_2 realizes M_3 then M_1 realizes M_3, i.e. realization is a transitive relation on the class of all sequential machines.

If (M', q') and (M, q) are initial state machines, then (M', q') realizes (M, q) if M' realizes M under a triple $(\sigma_1, \sigma_2, \sigma_3)$ and $\sigma_2(q) = q'$.

When the state behavior is to be simulated as well as the input-output behavior, this type of simulation is best described by the notion of homomorphic realization.

Definition 2.2

M' homomorphically realizes M if there is a triple of functions (η_1, η_2, η_3) where $\eta_1 : I^+ \longrightarrow I'^+$ where $\eta_1(I) \subseteq I'$, $\eta_2 : Q'' \longrightarrow Q$ (onto) where $Q'' \subseteq Q'$, $\eta_3 : Z'' \longrightarrow Z$ where $Z'' \subseteq Z'$, such that, for all $q \in Q''$ and $x \in I^+$, i) $\alpha_{\eta_2(q)}(x) = \eta_2(\alpha'_q(\eta_1(x)))$, and ii) $\beta_{\eta_2(q)}(x) = \eta_3(\beta'_q(\eta_1(x)))$.

The connection between realizations and homomorphic realizations can be stated precisely as follows [14]:

Theorem 2.2

If M and M' are sequential machines and M_R is a reduced machine equivalent to M, then M' realizes M if and only if M' homomorphically realizes M_R.

Therefore any realization of a reduced machine M_R is, in fact, a homomorphic realization of M_R. It simulates both the input-output behavior and the state behavior of M_R.

Given a machine M, two inputs a, b ϵ I are equivalent if $\forall q \epsilon Q$, $\lambda(q, a) = \lambda(q, b)$ and $\delta(q, a) \equiv \delta(q, b)$. A machine is transition distinct if no two inputs are equivalent. Any machine that has equivalent inputs is redundant in the sense that the inputs in an equivalent class can be represented by one of its member without affecting the functional capability of the machine.

We are to show in the following theorem that inputs of a given machine which maps into the same input of a realizing machine must be equivalent.

Theorem 2.3

If M' realizes M under $(\sigma_1, \sigma_2, \sigma_3)$ and a, b ϵ I such that $\sigma_1(a) = \sigma_2(b)$ then a and b are equivalent inputs.

Proof:

Since $\sigma_1(a) = \sigma_1(b)$, we have $\forall x \epsilon I^*$ and $\forall q \epsilon Q$, $\sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(ax))) = \sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(bx)))$. Hence, by the definition of realization,

$$\beta_q(ax) = \beta_q(bx) \qquad \forall x \epsilon I^*,$$

which implies

$$\lambda(q, a) = \lambda(q, b) \text{ and } \delta(q, a) \equiv \delta(q, b).$$

Thus a and b are equivalent inputs.

Corollary 2. 3. 1

If M' realizes a transition distinct machine M under $(\sigma_1, \sigma_2, \sigma_3)$ and $|I'| = |I|$, then σ_1 is a one-to-one and onto function.

Proof:

Immediate from Theorem 2. 3.

A similar result for states also holds.

Theorem 2. 4

If M' realizes M under $(\sigma_1, \sigma_2, \sigma_3)$ and $r, s \in Q$ such that $\sigma_2(r) = \sigma_2(s)$, then $r \equiv s$.

Proof:

Let x be any input sequence in I^+, then

$$\beta'_{\sigma_2(r)}(\sigma_1(x)) = \beta'_{\sigma_2(s)}(\sigma_1(x)).$$

So

$$\sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x)) = \sigma_3(\beta'_{\sigma_2(s)}(\sigma_2(x))).$$

But by definition M' realizes M under $(\sigma_1, \sigma_2, \sigma_3)$,

$$\beta_r(x) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x))).$$

Hence

$$\beta_r(x) = \beta_s(x) \quad \forall\, x \in I^+,$$

or $r \equiv s$.

Section 2.2 Checkable Realizations with Enlarged Input Sets

Given a noncheckable machine, can we find a checkable realization of the given machine with the same numbers of states and outputs, and with possibly a larger input set? The following theorem establishes a positive answer to this question.

Theorem 2.5

If (M, q) is a noncheckable machine, then it has a checkable realization (M', q') with $|I'| = |I| + 1$, $|Q'| = |Q|$, and $|Z'| = |Z|$.

Proof:

Let (M', q) be a machine such that $I' = I \cup \{a\}$, where $a \notin I$, and $M|I$ is identical to M. More precisely: $Q' = Q$, $Z' = Z$, and, $\forall\, r \in Q$ and $b \in I$, $\delta'(r, b) = \delta(r, b)$ and $\lambda'(r, b) = \lambda(r, b)$. Let $M' |a$ be be any strongly connected autonomous machine.

Therefore (M', q) is strongly connected, and hence [18] is checkable. We must also show that (M', q) is a realization of (M, q). Let $\sigma_1 : I^+ \longrightarrow (I')^+$, $\sigma_2 : Q \longrightarrow Q'$, and $\sigma_3 : Z' \longrightarrow Z$ all be identity functions. It follows that, $\sigma_2(q) = q$ and, $\forall\, r \in Q$ and $x \in I^+$,

$$\beta_r(x) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x))).$$

Therefore, by definition, (M', q) is a realization of (M, q). Clearly $|I'| = |I| + 1$, $|Q'| = |Q|$, and $|Z'| = |Z|$. This completes the proof.

When the given machine is not transition distinct, a checkable realization can be obtained even without augmenting the structure of the given machine.

Theorem 2.6

If (M, q) is a noncheckable machine which is not transition distinct, then it has a checkable realization with $|I'| = |I|$, $|Q'| = |Q|$, and $|Z'| = |Z|$.

Proof:

Since M is not transition distinct, these are equivalent inputs, say a and b $(a \neq b)$, in I. Let $M' = (I, Q, Z, \delta', \lambda')$ be a machine such that $M'|I - \{b\}$ is identical to $M|I - \{b\}$, and $M'|b$ is any strongly connected machine. Therefore (M', q) is strongly connected, and hence [18] is checkable. Let $\sigma_1 : I^+ \longrightarrow (I')^+$ be defined as, $\forall x \in I$, $\sigma_1(x) = x'$, where x' is the input sequence x with every b in it replaced by an a. Let $\sigma_2 : Q \longrightarrow Q'$ and $\sigma_3 : Z' \longrightarrow Z$ be identity functions. Since a and b are equivalent inputs, by definition, $\forall x \in I^+$ and $\forall r \in Q$, $\beta_r(x) = \beta_r(x') = \beta_r(\sigma_1(x))$. Clearly then, $\sigma_2(q) = q$ and, $\forall r \in Q$ and $x \in I^+$,

$$\begin{aligned} \beta_r(x) &= \beta_r(\sigma_1(x)) \\ &= \sigma_3(\beta_{\sigma_2(r)}(\sigma_1(x))). \end{aligned}$$

Thus, (M', q) is a checkable realization of (M, q) with $|I'| = |I|$, $|Q'| = |Q|$, and $|Z'| = |Z|$.

Therefore structure augmentations are not necessary for obtaining checkable realizations for machines that have equivalent inputs.

In the next three sections, the question of realizing a given non-checkable machine by a checkable machine will be studied in terms of three means of augmenting the structure of the given machine, i. e. of augmenting the input set, state set, or output/state sets of the given machine.

Section 2.3 Checkable Realizations with Enlarged State Sets

This section studies the question of whether it is possible to obtain a checkable realization of a given noncheckable machine by augmenting only the state set of the given machine. When the situation imposes more rigid restrictions on the number of input and output terminals than on the number of states of a system, such as in designing LSI circuit chips, a positive answer to the above question would be welcomed. Unfortunately, it turns out that any noncheckable machine which is structurally minimal does not have any checkable realization with only a larger state set. That is,

Theorem 2.7

Let (M, q) be reduced, transition-distinct, and noncheckable. And let the range of λ be Z, i.e. $\lambda(Q, I) = Z$. If (M', q') is a realization of (M, q), $|I'| = |I|$, and $|Z'| = |Z|$, then (M', q') is also noncheckable.

Proof:

Let (M', q') realize (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$. Then $\sigma_2(q) = q'$ and $\beta_r(y) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(y)))\ \forall\ r \in Q,\ y \in I^+$. Since M is transition-distinct and $|I'| = |I|$, by Corollary 2.3.1, σ_1 is 1-1 and onto. Hence, for each $u \in I^+$, there exists a unique input sequence in $(I')^+$, denoted u', such that $u' = \sigma_1(u)$. Since $|Z'| = |Z|$ and $\lambda(Q, I) = Z$, σ_3 is also 1-1 and onto.

Since σ_3 is 1-1, this implies

$$(\hat{\beta}_2)_{q_1}(x') = \hat{\beta}'_{q'}(x').$$

By assumption M is reduced, hence $|Q'| \geq |Q|$. Furthermore, $|Q_2| = |Q_1| \leq |Q|$. Therefore $|Q_2| \leq |Q'|$ and thus, by the construction of M_2, $M_2 \in \mathfrak{M}(M')$. Now since x' is a checking sequence for (M', q'), $M_2 \in \mathfrak{M}(M')$, $(\hat{\beta}_2)_{q_1}(x') = \hat{\beta}'_{q'}(x')$, it follows from the definition of checking sequences, $\exists\, r_1 \in Q_2$, $(\beta_2)_{r_1} = \beta'_{q'}$. Accordingly, $\forall\, y' \in (I')^+$,

$$\sigma_3((\hat{\beta}_2)_{r_1}(y')) = \sigma_3(\hat{\beta}'_{q'}\ (y'))$$

or

$$\sigma_3((\hat{\beta}_2)_{r_1}(\sigma_1(y))) = \sigma_3(\hat{\beta}'_{\sigma_2(q)}(\sigma_1(y))).$$

Thus

$$(\hat{\beta}_1)_{r_1}(y) = \beta_q(y) \qquad \forall\, y \in I^+,$$

or $r_1 \in Q_1$ and $r_1 \equiv q$, which says that x is a checking sequence for (M, q), a contradiction.

As a consequence of Theorem 2.7, the only remaining alternative to obtain a checkable realization of a machine without augmenting the input set would be to augment the output set.

Section 2.4 Checkable Realizations with Enlarged Output/State Sets

In this section, we will study the existence of checkable realizations of a given machine with enlarged output sets and possibly enlarged state sets.

First note that a simply connected machine is detectable if all its transitions have distinct outputs.

Theorem 2.8

If (M, q) is a simply connected machine such that all its transitions have distinct outputs, i.e., $\lambda(q, a) \neq \lambda(r, b)$ $\forall$ $q, r \in Q$ $(q \neq r)$ and $\forall$ $a, b \in I$ $(a \neq b)$, then (M, q) is detectable.

Proof:

If (M, q) is autonomous then it is detectable from [18], so let $|I| \geq 2$. We will construct a checking sequence for (M, q). Let (M, q) be simply connected with k components $C_1, C_2, \ldots, C_k$. For each component C_i, let q_i and r_i denote the initial state and the terminal state of C_i respectively. Hence $q_1 = q$ and r_k does not exist. Let a_i denote the input associated with the transition from r_i to q_{i+1}. By assumption, (M, q) is simply connected, so there is at most one transition leaving C_i, and since $|I| \geq 2$, $\exists$ $a \in I$ $\ni$ $\delta(r, a) \in C_i$ $\forall$ $r \in C_i$. An input sequence x_i can be constructed so that (1) x_i includes $y_{r,b} = au_rba$ for all $r \in C_i$, $b \in I$, and $(r, b) \neq (r_i, a_i)$, where u_r is a transfer input sequence which brings M to r, (2) $\alpha_{q_i}(x_i) = r_i$, and (3) when x_i is applied to (M, q_i), M will be in the same state of C_i just before $y_{r,b}$ for all $b \in I$.

Suppose, contrary to the theorem, that (M', q') has a checking sequence x'. We will prove the theorem by showing that $x(x' = \sigma_1(x))$ must be a checking sequence for (M, q), a contradiction.

Let (M_1, q_1) be a machine such that $M_1 \in \mathfrak{M}(M)$ and $(\hat{\beta}_1)_{q_1}(x) = \hat{\beta}_q(x)$. Let (M_2, q_1) be a machine such that $I_2 = I'$, $Q_2 = Q_1$, $Z_2 = Z'$, and δ_2 and λ_2 are defined as, $\forall\ s \in Q_2$ and $a' \in I_2 (= I')$,

$$\delta_2(s, a') = \delta_1(s, a)$$

and

$$\sigma_3(\lambda_2(s, a')) = \lambda_1(s, a).$$

(λ_2 can be uniquely defined in this way because σ_3 is 1-1 and onto). Clearly then, for all $s \in Q_1$ and $y \in (I_1)^+$,

$$(\hat{\beta}_1)_s(y) = \sigma_3((\hat{\beta}_2)_s(y')).$$

Thus (M_2, q_1) realizes (M_1, q_1) under $(\sigma_1, 1, \sigma_3)$, where 1 stands for the identity function. Hence

$$(\hat{\beta}_1)_{q_1}(x) = \sigma_3((\hat{\beta}_2)_{q_1}(x')),$$

and, since $(\hat{\beta}_1)_{q_1}(x) = \hat{\beta}_q(x)$,

$$\hat{\beta}_q(x) = \sigma_3((\hat{\beta}_2)_{q_1}(x')).$$

But $\hat{\beta}_q(x) = \sigma_3(\hat{\beta}'_{q'}(x'))$. Thus

$$\sigma_3((\hat{\beta}_2)_{q_1}(x') = \sigma_3(\hat{\beta}'_{q'}(x')).$$

The u_r mentioned in (1) always exists because the state of M before $y_{r,b}$ is in C_i, as required by (3), and thus the state of M between the u and the u_r of $y_{r,b}$ as well as the state r are both in C_i.

We claim that $x = x_1 a_1 x_2 a_2 \ldots a_{k-1} x_k$ is a detecting sequence for (M, q). Let (M', q') be a machine, where $M' \in \mathfrak{M}(M)$. If $\hat{\beta}'_{q'}(x) = \hat{\beta}_q(x)$, then because there are $|Q|$ distinct responses to the same input, say b, after the u_r or $y_{r,b}$ $\forall r \in Q$, $|Q'| \geq |Q|$. But $M' \in \mathfrak{M}(M)$ implies $|Q'| \leq |Q|$. Hence $|Q'| = |Q|$ and M' is reduced. Furthermore, $\hat{\beta}'_{q'}(x) = \hat{\beta}_q(x)$ implies that, for each $r \in Q$, the responses to au_r's in $y_{r,b}$ are identical, hence the states of M' after au_r's are the same. Let r' denote the state of M' after each u_r. For each $b \in I$, $y_{r,b}$ together with the corresponding response of (M', q') uniquely determine the output and the terminal state of the transition from r' to $\delta'(r', b)$ induced by b. This output is clearly identical to that of the transition of M from r to $\delta(r, b)$ induced by the same input b. Hence k components, denoted $C'_1, C'_2, \ldots, C'_k$, can be constructed from the responses of (M', q') to $x_1, x_2, \ldots, x_k$ in x. For each i, the terminal state r'_i of C'_i, the output $\lambda'(r'_i, a_i)$, and the initial state q'_{i+1} of C'_{i+1} can be determined from the responses of M to x_i, a_i and x_{i+1} respectively. Clearly $M' \equiv M$ and $q' \equiv q$, hence x is a detecting sequence for (M, q), and the theorem is established.

Example 2. 1

All the transitions of the machine (M, q) as shown in Figure 2. 2 have distinct ouputs. (M, q) has three components C_1, C_2 and C_3. As

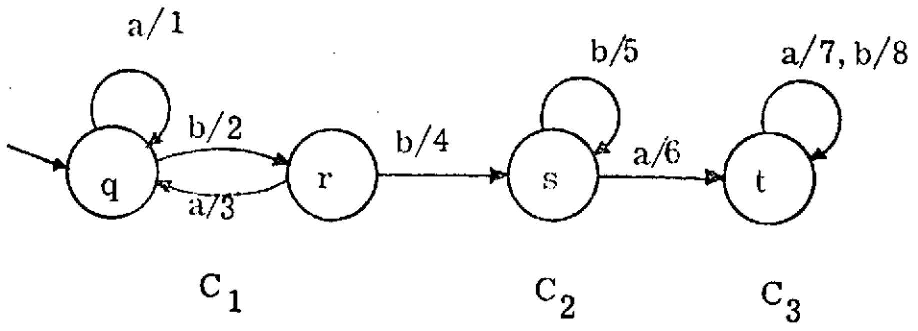


Figure 2. 2 Machine (M, q) for Example 2. 1

for C_1, it is clear that $a \in I$ is such that $\delta(r, a) \in C_1$, $\forall\, r \in C_1$. Hence we may choose the following sequences:

$$y_{q,a} = au_q aa$$

$$\underset{q}{\uparrow} y_{q,b} = au_q ba, \quad \text{where } u_q = \Lambda,$$

$$\underset{q}{\uparrow} y_{r,a} = au_r aa, \quad \text{where } u_r = b,$$

and

$$\begin{array}{lllllllllll} x_1 = & \underset{q}{\uparrow} & \overline{a\ a\ a} & b & a & b & a & a & \underset{r}{\uparrow} & b \\ & & y_{q,a} & & & & & & & \\ & & & \overline{\ \ \ \ \ \ \ } & & & & & & \\ & & & y_{q,b} & & & & & & \\ & & & & & \overline{\ \ \ \ \ \ \ } & & & & \\ & & & & & y_{r,a} & & & & \end{array}$$

where each arrow indicates the state of M at the time before the next input symbol, i. e., the next one on the right, is applied. As for C_2, since $\delta(s, b) \in C_2$, $\forall\, s \in C_2$, we may choose the following sequences:

$$\underset{s}{\uparrow} y_{s,b} = b\, u_s\, b\, b, \quad u_s = \Lambda$$

and

$$x_2 = \underset{s}{\uparrow}\underbrace{b\,b\,b}_{y_{s,b}}\underset{s}{\uparrow} \quad .$$

As for C_3, since it is strongly connected, we may choose the following sequences:

$$y_{t,a} = au_t aa$$

$$\underset{t}{\uparrow}\; y_{t,b} = au_t ba, \quad \text{where } u_t = \Lambda$$

and

$$x_3 = \underset{t}{\uparrow}\underbrace{a\,a\,\mathrlap{\underbrace{}_{\rule{0pt}{1.5em}y_{t,b}}}a}_{y_{t,a}}\,b\,a\underset{t}{\uparrow} \quad .$$

The interested reader is urged to verify that the sequence $x = x_1bx_2ax_3$ is indeed a checking sequence for (M, q).

It is clear that only the reachable portion of an initial-state machine plays a role in the normal operations of the machine. Hence in the following investigations we will concern mainly on realizing machines that are reachable.

Theorem 2.9

Let (M, q) be a reachable and transition distinct machine. If (M_{β_q}, β_q) is simply connected then (M, q) has a checkable realization (M', q') such that $|I'| = |I|$ and $|Q'| = k$, for any $k \geq |Q_{\beta_q}|$.

Proof:

Since all autonomous machines are checkable, this theorem is trivially true for autonomous machines. So suppose (M, q) is reachable, transition distinct and $|I| \geq 2$. We will construct a machine (M', q') in the following steps:

1) Set $(M', q') = (M_{\beta_q}, \beta_q)$. Since (M', q') is simply connected, let $C_1, C_2, \ldots, C_\ell$ be its linearly ordered components. Since C_ℓ is a strongly connected submachine of M and $|I| \geq 2$, we can [18] add $j = k - |Q_{\beta_q}|$ states $r_1, r_2, \ldots, r_j$ to C_ℓ to obtain a $(|C_\ell| + j)$-state strongly connected machine C'_ℓ which is equivalent to C_ℓ. After this is done, clearly, (M', q') is still simply connected and its initial-state behavior is unchanged.
2) Reassign outputs on all transitions of (M', q') so that they are all distinct.

Clearly, by Theorem 2.8, (M', q') is checkable. We need only to show that (M', q') is a realization of (M, q).

Let $(\sigma_1, \sigma_2, \sigma_3)$ be defined as follows:

$\sigma_1 : I \longrightarrow I'$ be an identity function;

$\sigma_2 : Q \longrightarrow Q'$ be such that

$\forall r \in Q, \ \sigma_2(r) = \beta_r \in Q'$;

and $\sigma_3 : Z' \longrightarrow Z$ be such that

$\forall \ b_{sa} \in Z'$ (where b_{ra} denotes the output on the transition of M' from the state s to the state $\alpha'_s(a)$

induced by the input a),

$$\sigma_3(b_{sa}) = \beta'_s(a).$$

Now, $\forall\, x \in I^+$ (let $x = yc$, $c \in I$) and $\forall\, r \in Q$,

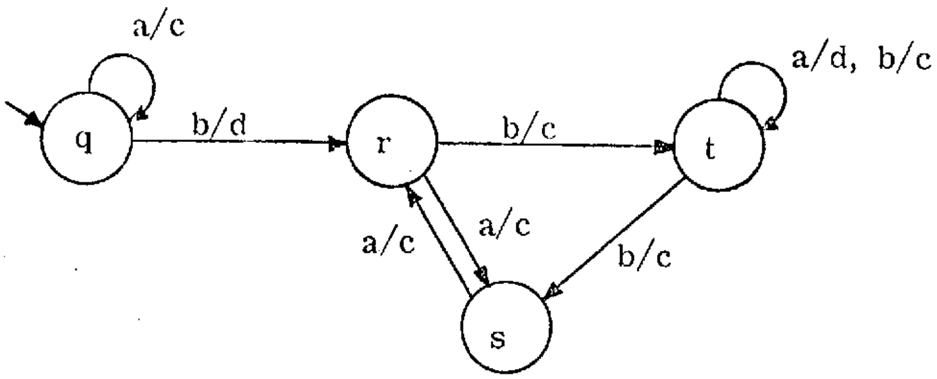
$$\begin{aligned}
\sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x)) &= \sigma_3(\beta'_{\sigma_2(r)}(x)) \\
&= \sigma_3(\beta'_{\beta_r}(x)) \\
&= \sigma_3(\beta'_{\beta_s}(c)), \text{ where } \beta_s = \alpha'_{\beta_r}(y) \\
&= \sigma_3(b_{\beta_s c}) \\
&= \beta'_{\beta_s}(c) \\
&= \beta_s(c) \\
&= \beta_r(x) .
\end{aligned}$$

Clearly $\sigma_2(q) = \beta_q = q'$. Therefore (M', q') realizes (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$. This completes the proof.

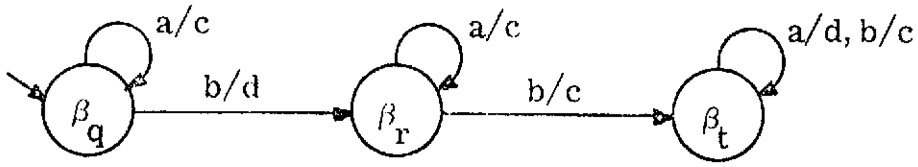
Example 2. 2

The machine (M, q) as shown in Figure 2. 3 is transition-distinct and reachable. The canonical machine (M_{β_q}, β_q) of (M, q) is simply connected and noncheckable. Following the procedure stated in the proof of Theorem 2. 9, a five-state checkable realization of (M, q) is obtained as the machine (M', β_q).

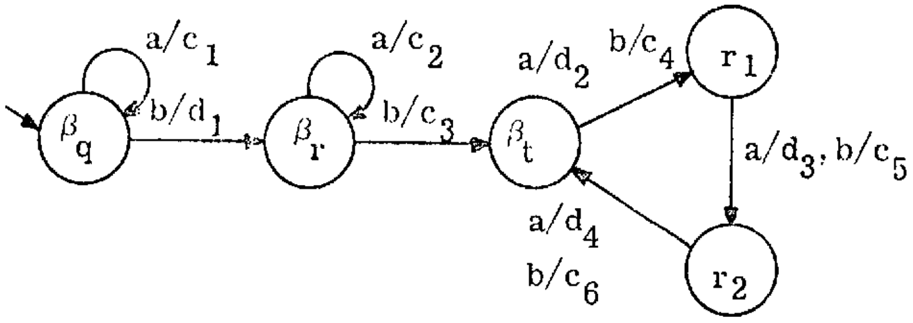
(M, q):



(M_{β_q}, β_q):



(M', β_q):



$\sigma_1: I^+ \longrightarrow (I')^+$ is an identity function,

$\sigma_2: Q \longrightarrow Q'$ is defined as, $\forall\ r \in Q$,

$$\sigma_2(r) = \beta_r$$

$\sigma_3: Z' \longrightarrow Z$ is defined as

$$\sigma_3(c_1) = \sigma_3(b_{\beta_q a}) = \beta'_{\beta_q}(a) = \beta_q(a) = c$$

Similarly, $\sigma_3(c_i) = c \quad \forall\ c_i$

and $\sigma_3(d_j) = d \qquad \forall\ d_j$

Figure 2.3 Machines for Example 2.2

Thus when the canonical form of a given reachable and transition distinct machine is simply connected, Theorem 2.9 says that we can always find a checkable realization for the given machine with no more inputs and virtually any specified number of states. Since the existence of such a realization is the focal issue in Theorem 2.9, we have made no attempt to minimize the number of outputs.

Our next aim will be to study the converse of the Theorem 2.9. We note first of all that:

<u>Theorem 2.10</u>

If (M', q') realizes (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$ then $(M_{\beta'_{q'}}, \beta'_{q'})$ realizes $(\overline{M, q})$, the reachable machine from q, under $(\sigma_1, \sigma'_2, \sigma_3)$ such that $\sigma'_2(r) = \beta'_{\sigma_2(r)}$ $\forall\, r \in R(q)$.

<u>Proof:</u>

Let $M_2 = M_{\beta'_{q'}} = (I, Q_{\beta'_{q'}}, Z, \delta_2, \lambda_2)$, $(M_3, q) = (\overline{M, q})$ and $M_3 = (I, Q_3, Z, \delta_3, \lambda_3)$. Hence $Q_3 = R(q)$, $\delta_3 = \delta \mid Q_3 \times I$, $\lambda_3 = \lambda \mid Q_3 \times I$. Now $\forall\, x \in I^+$

$$\sigma_3((\beta_2)_{\sigma'_2(r)}(\sigma_1(x)) = \sigma_3((\beta_2)_{\beta'_{\sigma_2(r)}}(\sigma_1(x)))$$

$$= \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x))).$$

But $\beta_r(x) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x))),$

hence $\sigma_3((\beta_2)_{\sigma_2(r)}(\sigma_1(x))) = \beta_r(x)$

$$= (\beta_3)_r(x) \quad \forall\, r \in Q_3.$$

Thus $(M_{\beta'_{q'}}, \beta'_{q'})$ realizes $(\overline{M, q})$ under $(\sigma_1, \sigma'_2, \sigma_3)$.

Corollary 2.10.1

If (M', q') realizes (M, q) then $(M_{\beta'_{q'}}, \beta'_{q'})$ realizes (M_{β_q}, β_q).

Proof:

Follows immediately from Theorem 2.2 and Theorem 2.10.

If M' realizes M under $(\sigma_1, \sigma_2, \sigma_3)$, then, $\forall\, r \in Q$, $\sigma_2(r)$ of Q' simulates r. It is possible that, in Q', there are other states that also simulate r. In order to describe precisely the relations among such states, we define a relation τ on $Q' \times Q'$ as $\tau = \{(r, t) \epsilon Q' \times Q' \mid \sigma_3(\beta'_r(\sigma_1(x))) = \sigma_3(\beta'_t(\sigma_1(x)))\ \forall\, x \in I^+\}$. We will use $r\tau t$ to denote $(r, t) \in \tau$. It can be easily verified that τ is an equivalence relation, and that $\forall\, s_1, s_2 \in Q$, $\sigma_2(s_1)\tau\sigma_2(s_2)$ implies $s_1 \equiv s_2$.

Theorem 2.11

If M' realizes M under $(\sigma_1, \sigma_2, \sigma_3)$ then for each transition $r_1 \xrightarrow{a/b} r_2$ of M there exists a transition $r'_1 \xrightarrow{\sigma_1(a)/b'} r'_2$ of M' such that $r'_1\tau\sigma_2(r_1)$, $\sigma_3(b') = b$ and $r'_2\tau\sigma_2(r_2)$.

Proof:

Let $r'_1 = \sigma_2(r_1)$, $b' = \beta'_{\sigma_2(r_1)}(\sigma_1(a))$, and $r'_2 = \alpha'_{\sigma_2(r_1)}(\sigma_1(a))$. Then, clearly $r'_1 \xrightarrow{\sigma_1(a)/b'} r'_2$ is a transition of M'. We claim that this transition satisfies the theorem. Since τ is an equivalence relation, $r'_1\tau r'_1$ and hence $r'_1\tau\sigma_2(r_1)$. By the definition of b',

$\sigma_3(b') = \sigma_3(\beta'_{\sigma_2(r_1)}(\sigma_1(a)))$. It follows from the definition of realization that $\sigma_3(\beta'_{\sigma_2(r_1)}(\sigma_1(a))) = \beta_{r_1}(a)$. Hence $\sigma_3(b') = \beta_{r_1}(a) = b$. It remains to be shown that $r'_2 \tau \sigma_2(r_2)$. Since, by the definition of input-output behavior functions, $\forall\, x \in I^+$,

$$\beta'_{r'_1}(\sigma_1(ax)) = \beta'_{\alpha'_{r_1}(\sigma_1,(a))}(\sigma_1(x)) = \beta'_{r'_2}(\sigma_1(x)).$$

Thus

$$\sigma_3(\beta'_{r'_1}(\sigma_1(ax))) = \sigma_3(\beta'_{r'_2}(\sigma_1(x))).$$

And, by the definition of realization and the fact that $r'_1 = \sigma_1(r_1)$,

$$\sigma_3(\beta'_{r'_1}(\sigma_1(ax))) = \beta_{r_1}(ax),$$

hence

$$\beta_{r_1}(ax) = \sigma_3(\beta'_{r'_2}(\sigma_1(x))),$$

or

$$\beta_{r_2}(x) = \sigma_3(\beta'_{r'_2}(\sigma_1(x))),$$

which means $r'_2 \tau \sigma_2(r_2)$. Thus the proof is complete.

When a realization machine is reachable and the input encoding function is onto, a theorem which is the converse of the above theorem can be established.

<u>Theorem 2.12</u>

If (M', q') realizes (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$ such that σ_1 is onto and (M', q') is reachable, then for each transition $r'_1 \xrightarrow{a'/b'} r'_2$ of M'

there is a transition $r_1 \xrightarrow{a/b} r_2$ of M such that $\sigma_2(r_1)\tau r_1'$, $\sigma_1(a) = a'$, $b = \sigma_3(b')$ and $\sigma_2(r_2)\tau r_2'$.

Proof:

Since (M', q') is reachable, let $y' \in (I')^*$ be such that $\alpha'_{q'}(y') = r_1'$. The fact that (M', q') realizes (M, q) implies $\beta_r(x) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x)))$, $\forall\, r \in Q$ and $x \in I^+$, and $\sigma_2(q) = q'$. Furthermore, σ_1 is onto, so $\exists\, y \in I^+ \ni \sigma_1(y) = y'$. Let $r_1 = \alpha_q(y)$. Then

$$\begin{aligned}\beta_{r_1}(x) &= \beta_q(yx)\\ &= \sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(yx)))\\ &= \sigma_3(\beta'_{q'}(\sigma_1(yx)))\\ &= \sigma_3(\beta'_{r_1'}(x)) \qquad \forall\, x \in I^+.\end{aligned}$$

Hence

$$\sigma_2(r_1)\,\tau\, r_1'$$

Since σ_1 is onto, let $a \in I$ be such that $\sigma_1(a) = a'$, then

$$\begin{aligned}b = \beta_{r_1}(a) &= \sigma_3(\beta'_{r_1'}(\sigma_1(a)))\\ &= \sigma_3(\beta'_{r_1'}(a'))\\ &= \sigma_3(b').\end{aligned}$$

It remains to be shown that $\sigma_2(r_2)\tau r_2'$. For all $x \in I^+$,

$$\begin{aligned}\beta_{r_2}(x) &= \beta_q(yax)\\ &= \sigma_3(\beta'_{q'}(\sigma_1(yax)))\\ &= \sigma_3(\beta'_{\alpha'_{q'}(\sigma_1(ya))}(\sigma_1(x)))\\ &= \sigma_3(\beta'_{\alpha'_{q'}(y'a')}(\sigma_1(x))\\ &= \sigma_3(\beta'_{r'_2}(\sigma_1(x))).\end{aligned}$$

Thus $\sigma_2(r_2)\tau r'_2$.

Before we show how the simple connectivity of a machine is implied by the simple connectivity of its realizations, let us note this:

Theorem 2.13

Let (M', q') be a realization of (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$ such that σ_1 is onto. Furthermore, let (M', q') be reachable and (M, q) be reduced. If (s_1, s_2) is a binary partition of Q then (s'_1, s'_2) is a binary partition of Q', where

$$s'_1 = \{s' \epsilon Q' \mid \exists\, s \epsilon S_1 \text{ such that } s' \tau \sigma_2(s)\} \quad \text{and}$$

$$s'_2 = \{s' \epsilon Q' \mid \exists\, s \epsilon S_2 \text{ such that } s' \tau \sigma_2(s)\}.$$

Proof:

We must show that $S'_1 \cap S'_2 = \phi$ and $Q' = S'_1 \cup S'_2$. Suppose $S'_1 \cap S'_2 \neq \phi$, let $r' \epsilon S'_1 \cap S'_2$. Then there exist $s_1 \epsilon S_1$ and $s_2 \epsilon S_2$ such that $r' \tau \sigma_2(s_1)$ and $r' \tau \sigma_2(s_2)$. Hence $\sigma_2(s_1)\, \tau \sigma_2(s_2)$ or $s_1 \equiv s_2$. But (M, q) is reduced, so we must have $s_1 = s_2$ or $s_1 \epsilon S_1 \cap S_2$ which contradicts that (S_1, S_2) is a binary partition of Q. Thus $S'_1 \cap S'_2 = \phi$. Clearly $S'_1 \cup S'_2 \leq Q'$. Now let $r' \in Q'$, we will show that $r' \in S'_1 \cup S'_2$, and hence establish that $Q' = S'_1 \cup S'_2$. Since (M', q') is reachable, $\exists y' \in (I')^*$ such that $\alpha'_{q'}(y') = r'$. Because σ_1 is onto, there exists a $y \in I^*$ such that $\sigma_1(y) = y'$. Moreover, since (M', q') is a realization of (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$, by definition, $\sigma_2(q) = q'$. Therefore $\beta_{\alpha_q(y)}(x) = \beta_q(yx) = \sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(yx))) = \sigma_3(\beta'_{q'}(y'\sigma_1(x))) = \sigma_3(\beta'_{r'}(\sigma_1(x)))$, or $\sigma_2(\alpha_q(y)) \tau r'$. Since $\alpha_q(y) \in Q$, r' must be in either S'_1 or S'_2, or $r' \in S'_1 \cup S'_2$.

Applying the Theorems 2.11, 2.12, and 2.13, we are able to show the following:

Theorem 2.14

Let (M, q) be a reduced, reachable, and transition distinct machine. If (M', q') realizes (M, q) under $(\sigma_1, \sigma_2, \sigma_3)$, then (M, q) is simply connected if (M', q') is.

Proof:

Suppose, contrary of the theorem, that (M, q) is not simply connected. Then by Theorem 1.2, there is a binary partition $\{S_1, S_2\}$ of Q such that $q \in Q$ and there is more than one transition from S_1 to S_2 but none from S_2 to S_1. Let $r_1 \xrightarrow{a/b} r_2$ and $t_1 \xrightarrow{c/d} t_2$ be two such transitions, so $r_1 \neq t_1$ or $a \neq c$. Let $S'_1 = \{s' \in Q' \mid \exists s \in S_1 \text{ such that } \sigma_2(s) \tau s'\}$ and $S'_2 = \{s' \in Q' \mid \exists s \in S_2 \text{ such that } \sigma_2(s) \tau s'\}$. Since M is transition distinct, it follows from Corollary 2.3.1 that σ_1 is one-to-one and onto.

Then by Theorem 2.13, $\{S'_1, S'_2\}$ is a binary partition on Q'.

By Theorem 2.11, (M'q') has two transitions $r'_1 \xrightarrow{\sigma_1(a)/b} r'_2$ and $t'_1 \xrightarrow{\sigma_1(c)/d} t'_2$ such that $r'_1 \tau \sigma_2(r_1)$, $r'_2 \tau \sigma_2(r_2)$, $t'_1 \tau \sigma_2(t_1)$ and $t'_2 \tau \sigma_2(t_2)$. Clearly then r'_1 and t'_1 are in S'_1, and r'_2 and t'_2 are in S'_2. Furthermore, since $r_1 \neq t_1$ or $a \neq b$, M is reduced and transition distinct. By Theorem 2.3 and 2.4, we conclude that $r'_1 \neq t'_1$ or $\sigma_1(a) \neq \sigma_2(b)$. Therefore there are two transitions from S'_1 to S'_2. There can be no transition from S'_2 to S'_1, for if there were then, by Theorem 2.12, there would also be a transition from S_2 to S_1 which is not true. But this contradicts the hypothesis that (M', q') is simply connected. Hence (M, q) must be simply connected.

We are ready to show one of the major results of this section, which is stronger than the converse of Theorem 2.9.

Theorem 2.15

Let (M, q) be a transition distinct machine. If (M, q) has a checkable realization with the same number of inputs then (M_{β_q}, β_q) is simply connected.

Proof:

Let (M', q') be a checkable realization of (M, q) such that $|I'| = |I|$. Then $(M_{\beta'_{q'}}, \beta'_{q'})$ realizes (M_{β_q}, β_q) as implied by Corollary 2.10.1. Also, from [8], $(M_{\beta'_{q'}}, \beta'_{q'})$ must be simply connected. Since (M, q) is transition distinct, (M_{β_q}, β_q) should also be transition distinct. Moreover, $I_{\beta'_{q'}} = I'$ and $I_{\beta_q} = I$, hence $|I_{\beta'_{q'}}| = |I_{\beta_q}|$. By Corollary 2.3.1, σ_1 is therefore an onto function. In addition $(M_{\beta'_{q'}}, \beta'_{q'})$ and (M_{β_q}, β_q)

are reduced and reachable. Recalling that $(M_{\beta'_{q'}}, \beta'_{q'})$ is simply connected, we can apply Theorem 2. 14 and conclude that (M_{β_q}, β_q) is also simply connected.

If a given initial-state machine is both transition distinct and reachable, a necessary and sufficient condition as to when the given machine has a checkable realization with no more inputs can be obtained readily from Theorem 2. 9 and Theorem 2. 15.

Theorem 2. 16

Let (M, q) be a transition distinct and reachable machine. Then (M, q) has a checkable realization with no more inputs if and only if (M_{β_q}, β_q) is simply connected.

Section 2. 5 Summary

In this chapter, we have investigated the problem of improving one aspect of the diagnosability, i. e. the checkability, of machines by machine structural augmentation and machine realization. The existence of checkable realizations for a given machine has been studied in three finer categories: they are checkable realizations that have (1) a larger input set, (2) a larger state set, and (3) a larger output set with possibly a larger state set (all relative to that of the given machine). The answer to the existence of such realizations has been found to be positive for the case 1) and 3), and negative for the case 2). More specifically, for a given machine M, there always exists a checkable realization M' such that M' has one more input symbol than M. If (M, q) is transition distinct and reachable, then there exists a checkable realization (M', q') with the same number of inputs if and only if (M_{β_q}, β_q), i. e., the canonical realization of (M, q), is simply connected.

III. MACHINE AUGMENTATIONS WITH REPEATED SYMBOL DISTINGUISHING SEQUENCES

Section 3.1 Introduction

Traditional system diagnosis often involves placing input and/or output test leads at various interconnections of the system in order to facilitate and speedup the testing process. However, interconnections of system components are not usually accessible; even if they are accessible, test points are hard to select and thus make system diagnosis a difficult feat. One way to overcome this difficulty is to design the system so that, using (only) the system input/output terminals, efficient diagnostic tests can be performed without adding any extra test leads.

This chapter concerns how to design machines with efficient checking sequences. In particular, the question of how to find an economical realization that has a repeated symbol distinguishing sequence (RDS) is studied. This problem is approached via a formal notion of machine augmentation. An augmentation of a machine is a realization such that its input, state and/or output set include that of the given machine. In particular, an input augmentation is an augmentation such that its input set properly includes that of the given machine. Similar meanings are linked to the state, output, or state-output augmentations. With respect to augmentations with RDS, input augmentations exist for any machine [10], [11] and it will be shown that state-augmentations do not exist for any reduced machine that does not have a distinguishing sequence. On the other hand, output augmentations generally result in corresponding circuit realizations having an excessive increase in the number of output terminals. This is undesirable because a limited number of output pins are allowed for each LSI chip. Therefore this study is devoted wholly to state-output augmentations, in which moderate enlargement of the output set is attained at the expense of an enlarged state set.

Section 3.2 Machines with Repeated Symbol Distinguishing Sequences

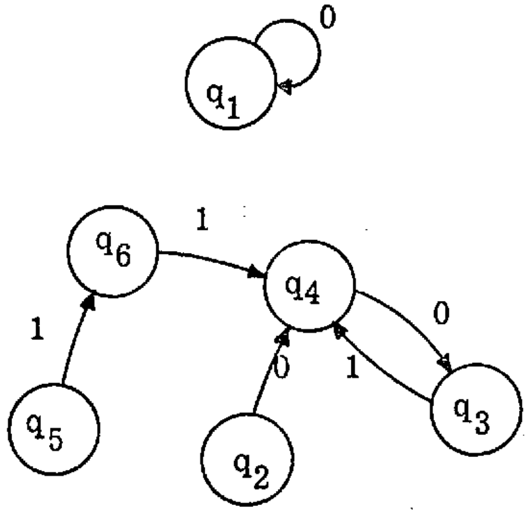
In this section, we will derive an upper bound on the length of an RDS and a characterization of the machine which possesses an RDS, both in terms of some structural properties of the state graph of the machine. Since an RDS of a machine is determined by a single-input submachine, we shall examine some properties of autonomous machines.

It can be seen that the graph of an autonomous machine consists of one or more weak components. Each of these weak components consists of exactly one semicycle, which in fact is a cycle, and some (or none) paths which terminate on the nodes of this cycle. We will call each such weak component of (the graph of) an autonomous machine a _flower._ In a flower, each node of the cycle which has paths (other than that of the cycle) terminating on it will be called a _root._ The subgraph that consists of all the paths (other than that of the cycle) terminating on a root of a flower will be called an _in-tree._ It can be easily seen that each node in an in-tree has a unique path to the root, and that an in-tree does not contain any semicycle. Summing up, the graph of an autonomous machine is a set of disconnected flowers, each flower consists of exactly one cycle (and no other semicycle) and some (or none) in-trees. For each node r in an in-tree T, the _branch in-tree at r_ is a subgraph of T which consists of all nodes (and arcs) that can reach r.

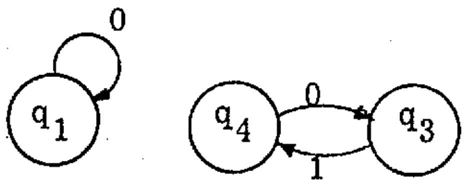
As an illustration, in Figure 3.1, D_o is the graph of M. Since inputs are all the same, they are eliminated from D_o.

	δ/λ
$Q \backslash I$	0
q_1	$q_1/0$
q_2	$q_4/0$
q_3	$q_4/1$
q_4	$q_3/0$
q_5	$q_6/1$
q_6	$q_4/1$

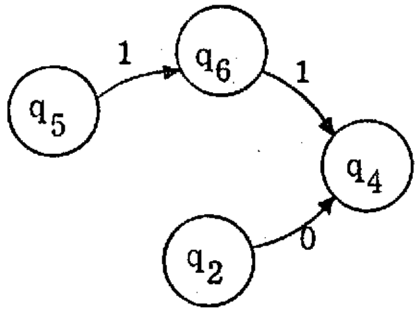
M



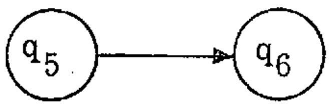
D_o



Two cycles



An in-tree with q_4 as its root



The branch in-tree at q_6

Figure 3. 1 The Graph for an Autonomous Machine

Let M be a sequential machine. We will use D_a to denote the graph of $M|a$, where $a \in I$; $C_q(a)$ to denote the cycle of the flower (in D_a) which contains q; $d_q(a)$ to denote the length of the path from q to the nearest node of $C_q(a)$, hence $d_q(a) = 0$ if q is on the cycle $C_q(a)$; $p_q(a)$ to denote the period of $C_q(a)$. When the input a is known, $C_q(a)$, $d_q(a)$ and $p_q(a)$ will be abbreviated as C_q, d_q and p_q.

In the graph of an autonomous machine, a cycle, a flower, or the graph itself will be viewed as the machines they correspond to. Hence a cycle is reduced if the autonomous machine it corresponds to is reduced. Two cycles are <u>equivalent</u> if their corresponding machines are equivalent. Two cycles are <u>similar</u> if they are equivalent and have the same period.

We will derive an upper bound on the length of an RDS of any given autonomous machine expressed in terms of machine graphical properties. We recall first of all that if q and r are non-equivalent states of an n-state machine, then they can be distinguished by a sequence of length $< n$.

Based on this, it has been shown that an autonomous machine has an RDS if and only if it is reduced [10]. Hence the characterization of reduced autonomous machines will be basic for our later study on finding realizations with RDS's.

Let D_a denote the graph of an autonomous machine with the input a.

Theorem 3.1

If q and r are states in two nonequivalent cycles C_q and C_r of D_a, then there is an integer $j < p_q + p_r$ such that q and r can be distinguished by a^j.

Proof:

Let $M' = (I, Q', Z, \delta|Q' \times I, \lambda|Q' \times I)$ where $Q' = \{q \mid q \text{ is in } C_q \text{ or } C_r\}$. Since $\beta_q \not\equiv \beta_r$, there is a sequence $x = a^j$ such that $j = \ell(a^j) < |Q'| = p_q + p_r$.

In the event the period of one cycle divides that of the other cycle, then states of nonequivalent cycles can be distinguished by a shorter input sequence.

Theorem 3.2

If q and r are in nonequivalent cycles C_q and C_r of D_a, C_q is reduced, and $p_r | p_q$, i.e., p_r divides p_q, then q and r can be distinguished by an input sequence a^ℓ where $\ell \leq p_q$.

Proof:

Let $p_q = kp_r$, $i = p_r$ and $j = p_q$. Suppose $\hat{\beta}_q(a^j) = \hat{\beta}_r(a^j) = b_{11}b_{12}\cdots b_{1i}b_{21}b_{22}\cdots b_{2i}\cdots b_{ki}\cdots b_{ki}$. Since i is the period of C_r, we must have $b_{ut} = b_{vt}$, for all $1 \leq u, v \leq k$ and $1 \leq t \leq i$. Let $q' = \alpha_q(a^i)$. Then $\hat{\beta}_{q'}(a^j) = b_{21}b_{22}\cdots b_{2i}\cdots b_{k1}b_{k2}\cdots b_{ki}b_{11}b_{12}\cdots b_{ki}$, hence $\hat{\beta}_{q'}(a^j) = \hat{\beta}_q(a^j)$, which implies $q' \equiv q$. But clearly $q' \neq q$. This contradicts that C_q is reduced.

Applying Theorem 3.1 and Theorem 3.2, an upper bound on the shortest length of RDS's can now be derived.

<u>Theorem 3.3</u>

If $M = (\{a\}, Q, Z, \delta, \lambda)$ is reduced, then M has a RDS of length $k \leq \max_{q \in Q} d_q + p_1 + p_2 - 1$, where p_1 is the period of the largest cycle in D, the graph of M, and p_2 is either the period of the second largest cycle in D such that $p_2 \nmid p_1$ or is equal to 1 if no such cycle exists.

<u>Proof:</u>

Let q and r be any two distinct states of Q. Let $d = \max\{d_q, d_r\}$, $q' = \alpha_q(a^d)$ and $r' = \alpha_r(a^d)$. Clearly q' and r' are states on C_q and C_r respectively. Assume without loss of generality that $p_q \geq p_r$. There are two cases to be considered.

Case 1: If $p_q \nmid p_r$, then clearly $p_1 \geq p_q$ and $p_2 \geq p_r$, hence $\max_{q \in Q} d_q + \ell_1 + \ell_2 - 1 \geq d + \ell_q + \ell_r - 1$. Since C_q and C_r are reduced and nonequivalent cycles, by Theorem 3.1, q' and r' can be distinguished by an input sequence of length $\ell_q + \ell_r - 1$. This implies that q and r can be distinguished by $a^{d+\ell_q+\ell_r-1}$.

Case 2: If $p_r | p_q$, then by Theorem 3.2, q and r can distinquished by a^{d+p_q}. Since, in this case, $\max_{q \in Q} d_q + \ell_1 + \ell_2 - 1 \geq \max_{q \in Q} d_q + \ell_1 \geq d + p_q$, the theorem is true.

A set of states S of a machine M <u>converges</u> under $a \in I$ if $\delta(q,a) = \delta(r,a)$ and $\lambda(q,a) = \lambda(r,a)$, $\forall\, q, r \in S$. A <u>k-convergence</u> at $q \in Q$ is a set S of size k that converges to the state q(under the same input a).

The necessary and sufficient condition for a flower to be reduced will be shown next.

<u>Theorem 3.4</u>

A flower F is reduced if and only if F is convergence-free and the cycle of F is reduced.

<u>Proof:</u>

(Necessity) Obviously true.

(Sufficiency) Suppose $q, r \in F$ and $q \equiv r$ but $q \neq r$. Since the cycle, denoted C, of F is reduced, q and r cannot both be on C. Let $xa \in I^+$ be an input sequence such that $\alpha_q(xa) \in C$, $\alpha_r(xa) \in C$ and $\alpha_q(x)$ or $\alpha_r(x) \notin C$. If $\alpha_q(x) \neq \alpha_r(x)$, then, since $q \equiv r$ implies $\alpha_q(xa) \equiv \alpha_r(xa)$ and $\beta_q(xa) = \beta_r(xa)$ and since C is reduced, we must have $\alpha_q(xa) = \alpha_r(xa)$ and $\beta_{\alpha_q(x)}(a) = \beta_{\alpha_r(x)}(a)$. Equivalently, $\alpha_q(x)$ and $\alpha_r(x)$ converge under a, which contradicts that F is convergence-free. On the other hand, if $\alpha_q(x) = \alpha_r(x)$, let yb be the shortest prefix of x such that $\alpha_q(y) \neq \alpha_r(y)$ and $\alpha_q(yb) = \alpha_r(yb)$. Since $q \equiv r$, $\beta_{\alpha_q(y)}(b) = \beta_{\alpha_r(y)}(b)$. Then $\alpha_q(y)$ and $\alpha_r(y)$ converge under b, again contradicts the fact that F is convergence-free. Hence no two states of F are equivalent and so F is reduced.

We are ready to establish a necessary and sufficient condition for an autonomous machine to be reduced.

Theorem 3. 5

Any autonomous machine M with input a is reduced if and only if its graph D is such that

i) D has no equivalent cycles,

ii) D is convergence-free, and

iii) all cycles in D are reduced.

Proof:

(Necessity) Obviously true.

(Sufficiency) By Theorem 3. 4 condition ii) and iii) ensure that all flowers in D are reduced. Hence we need only show that states on distinct flowers are distinguishable. Let q and r be two states on distinct flowers F_1 and F_2 of D respectively. Let $d = \max\{d_q, d_r\}$. Then $\alpha_q(a^d)$ and $\alpha_r(a^d)$ are states on C_q and C_r respectively since C_q and C_r are reduced and nonequivalent. From Theorem 3. 2, $\alpha_q(a^d) \not\equiv \alpha_r(a^d)$; hence $q \not\equiv r$ and M is reduced.

Section 3.3 Machine Augmentations

A machine specification given to a designer is usually a behavioral description which is equivalent to the description of an irredundant machine. Hence we may assume without loss of generality that, in our study to improve machine diagnosability, all machines under consideration are reduced and transition distinct. It can be shown easily that realizations for reduced and transition distinct machines have the same or larger input, state and output sets as that of the given machine. Hence such a realization can always be transformed into another realization whose input, state and output sets includes that of the given machine as subsets. One attraction about this type of realization is that when it is used to simulate the given machine only simple or trivial translations of the inputs and outputs are necessary. Thus devices that are used to perform translations are simple, and consequently, the cost of building one such realization is reduced. More importantly, the probability of failures in these devices is also smaller. To precisely describe such realizations, let M' and M be two sequential machines. Thus:

Definition 3.1

M' is an augmentation of M if M realizes M under $(\sigma_1, \sigma_2, \sigma_3)$ such that $I \subseteq I'$, $Q \subseteq Q'$, $Z \subseteq Z'$, and σ_1 and $\sigma_3 | Z$ are identity functions. Let M' be an augmentation of M. Then M' is an input augmentation if $I \subsetneq I'$, $Q = Q'$ and $Z = Z'$, (state augmentation and output augmentation are similarly defined); M' is a state-output augmentation if $I = I'$, $Q \subsetneq Q'$ and $Z \subsetneq Z'$.

For machines that are reduced and transition distinct, the notion of augmentation is just as general as that of realization. It should be clear that the problem of adding test input or output terminals to a sequential machine can be phrased in terms of input or output augmentation, but not vice versa in general. Various requirements as well as various diagnosabilities can be posed on augmentations to yield machines that have certain desired behavioral characteristics. In the following study, we will investigate the problem of how to find economical augmentations that possess RDS's.

Input augmentations with RDS's have been investigated previously and are known to exist [10], [11], hence will not be explored here. We will show that a state augmentation with RDS does not exist for a reduced and transition distinct machine which does not have an RDS.

Theorem 3. 6

Let M be a reduced and transition distinct machine and M' be a state augmentation of M under $(\sigma_1, \sigma_2, \sigma_3)$. If, $\forall$ a ϵ I, M$|$a is not reduced then, $\forall$ a' ϵ I', M'$|$a' is not reduced.

Proof:

Let a ϵ I. If M$|$a is not reduced, then there exist q, r ϵ Q $(q \neq r)$ such that $\beta_q(x) = \beta_r(x)$, $\forall$ x ϵ $\{a\}^+$. By the definition of realization, this implies that, $\forall$ x ϵ $\{a\}^+$,

$$\sigma_3(\beta'_{\sigma_2(q)}(\sigma_1(x))) = \sigma_3(\beta'_{\sigma_2(r)}(\sigma_1(x))).$$

Since M' is a state augmentation of M, we have I' = I and Z' = Z, and $\sigma_1 | I$ and $\sigma_3 | Z$ are identity functions, hence

$$\beta'_{\sigma_2(q)}(x) = \beta'_{\sigma_2(r)}(x) \quad \forall\, x \in \{a\}^+$$

But since M is reduced, $q \not\equiv r$. By Theorem 3. 3, we have $\sigma_2(q) \not\equiv \sigma_2(r)$, hence M''|a is not reduced. Because a is arbitrarily chosen and I' = I, $\forall\, a \in I'$, M' |a is not reduced.

Since a machine M has a RDS if and only if M|a is reduced, from Theorem 3. 6, we immediately obtain:

<u>Corollary 3. 6. 1</u>

Let M be reduced and transition distinct machine. If M has no RDS then no state augmentation of M has an RDS.

On the other hand output augmentations with RDS's generally result in corresponding circuit realizations having an excessive increase in the number of output terminals. This is undesirable in designing LSI circuits because a limited number of output pins are allowed for each LSI chip. Therefore, our study will be devoted to the state-output augmentation with RDS, in which the output set need only be enlarged moderately at the expense of an enlarged state set.

Section 3. 4 State-Output Augmentations with RDS's

Since the minimization of output sets is our primary concern in searching for state-output augmentations with RDS's, the approach we adopt is to minimize the output set of the state-output augmentation while placing no limit on the size of its state set. After the output set is minimized, we then seek ways to minimize the state set in later sections.

A cycle C_1 is said to be enlarged k times into another cycle C_2 if it is broken at any node into a sequence of arcs, and then joining k copies of that sequence to form a cycle C_2 with nodes renamed to avoid duplications. For example, in Figure 3. 2, the cycle on the left is enlarged twicefold into the cycle on the right.

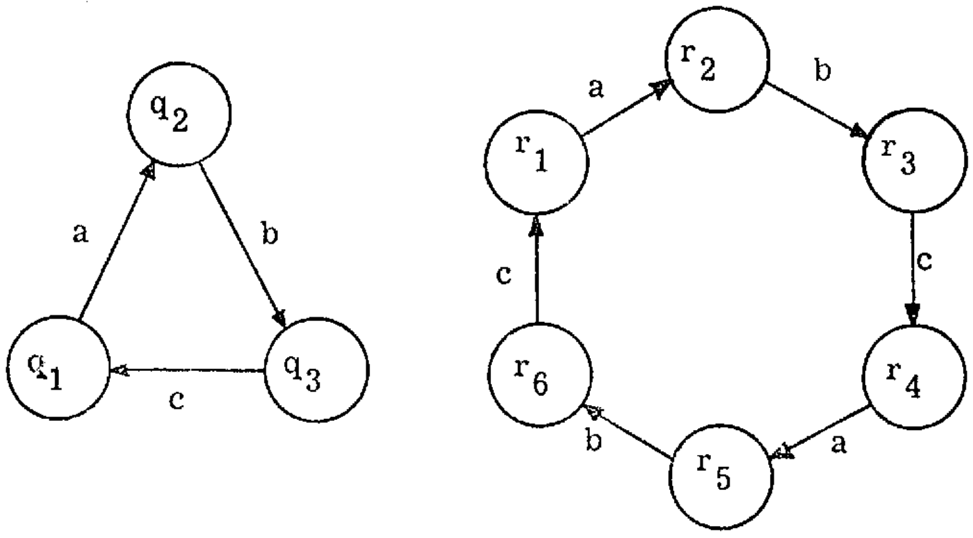


Figure 3. 2 A Cycle Is Enlarged Twice

Note that the cycle to be enlarged can be broken at any node, and the resulting enlarged cycle will always be equivalent to the original cycle.

Let D_a be the graph of an autonomous machine M with input a. A node $q \in D_a$ is a <u>source node</u> if there is no arc terminating on it. For each source node in D_a, we associate with it a subgraph G_q which consists of the path from q to the cycle $C_q(a)$ and the cycle.

We may now establish the theorem that asserts the existence, for any machine, of a state-output argumentation (with RDS) which has no more than twice as many output symbols as that of the given machine.

<u>Theorem 3.7</u> (Double-Z Augmentation)

Let M be a sequential machine with no RDS. Then M has a state-output augmentation M' with RDS such that $|Z'| \leq 2|Z|$.

<u>Proof:</u>

M does not have an RDS, so, $\forall a \in I$, $M|a$ is not reduced. We will construct another sequential machine M' in the following steps.

i) Get D_a, where a is any input in I.

ii) (To resolve convergences in in-trees of D_a). Skip this step if there is no convergence in the in-trees of D_a.

$$\text{Get } S = \{G_q \mid q \in Q \text{ is a source node in } D_a)\}$$
$$\cup \{\text{all flowers in } D_a \text{ that are cycles}\}.$$

Rename states in S such that, for each $q \in Q$, all q's in S have distinct names q, q^1, q^2, etc. Let $E'(q) = \{q, q^1, q^2, \ldots\}$, i.e. it is the set of all states in S that were q's. Hence, $\forall q \in Q$, states in $E'(q)$ are equivalent in S.

iii) (To force equivalent cycles to be dissimilar)

Construct S' by enlarging cycles in S so that no two equivalent cycles have the same period. Rename the states in S' such that, for each $q \in Q$, all q's and q^{i}'s in S' have distinct names, q, q_1, q_2, etc. Let $E(q) = \{q, q_1, q_2, \ldots\}$, i.e. it is the set of all states that are split from states of $E'(q)$. Hence, $\forall q \in Q$, states in $E(q)$ are equivalent in S'.

iv) (To resolve equivalent and nonreduced cycles)

Let $Z = \{b_1, b_2, \ldots, b_m\}$ and let $Z' = \{b'_1, b'_2, \ldots, b'_m\} \cup Z$, where $b_i \notin Z$, $b'_i \neq b'_j$ if $i \neq j$. In each cycle of S', if b_i is the output on the arc terminating on the root, then change b_i to b'_i; if there is no root, then change the output, say b_j, on any arc to b'_j.

v) (To embed $M'|a$ into M')

Let M' be a machine such that (1) the graph of $M'|a$ is S', and (2) $\forall r' \in Q'$ and $\forall c \in I$ $(c \neq a)$, $\delta'(r', c) = \delta(r, c)$ and $\lambda'(r', c) = \lambda(r, c)$, where $r \in Q$ and $r' \in E(r)$.

As a result of steps iii) and iv), all cycles in S' are reduced and nonequivalent. Moreover, since the in-trees in S' are all single paths, there can be no convergence in S' except, possibly, at the root of a cycle, but this is also ruled out in step iv) by the changes made on the output of the arc terminating on the root. Hence, by Theorem 3. 5, S' is reduced. Now let $\sigma_2: Q \longrightarrow Q'$ be the identity function, and let

$\sigma_3: Z' \longrightarrow Z$ be such that, $\forall b_i \epsilon Z_i$, $\sigma_3(b_i) = b_i$, and, $\forall b_i' \epsilon Z' - Z$, $\sigma_3(b_i') = b_i$. Clearly $\beta_r(x) = \sigma_3(\beta_r'(x)) \ \forall\ r \epsilon Q$, hence M' is a state-output augmentation of M with RDS.

In step v) of the above proof, the way of embedding M'$|$a into M' is certainly not unique. The transition function and output function can be defined in any fashion, as long as the machine M' obtained is a state-output augmentation of M including M'$|$a as a sub-machine.

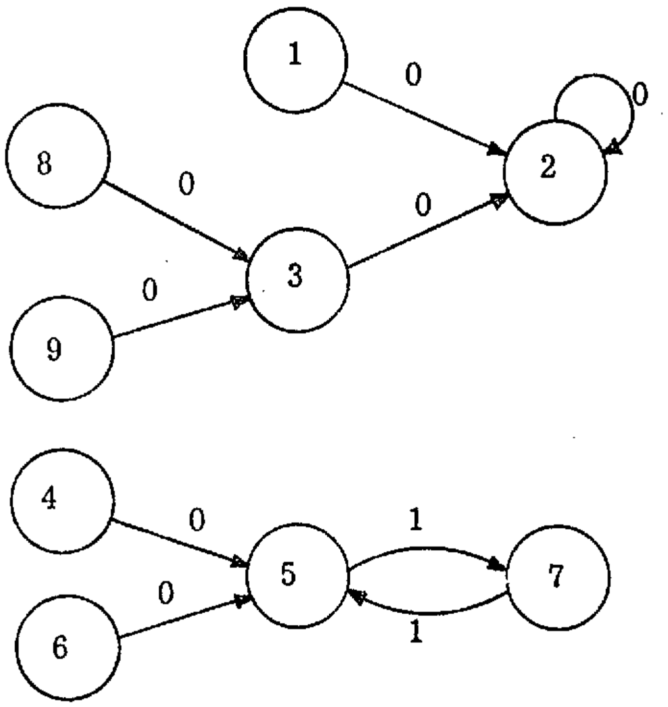
It is best to consider an example to illustrate the steps stated in the proof of Theorem 3.7. Let M be the machine in Figure 3.3.

	δ/λ	
Q \ I	0	1
1	2/0	9/0
2	2/0	1/1
3	2/0	4/1
4	5/0	1/1
5	7/1	8/0
6	5/0	1/1
7	5/1	6/0
8	3/0	9/0
9	3/0	8/0

Figure 3.3

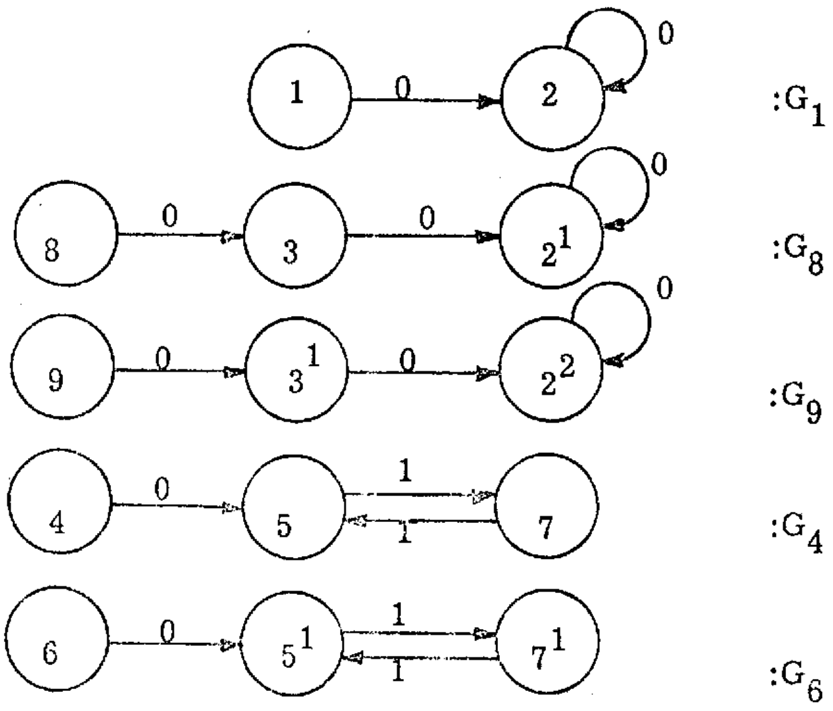
The steps to obtain M' as shown below will correspond to those in the proof of Theorem 3.7.

i) Choose 0ϵ I and get D_0.



: D_o

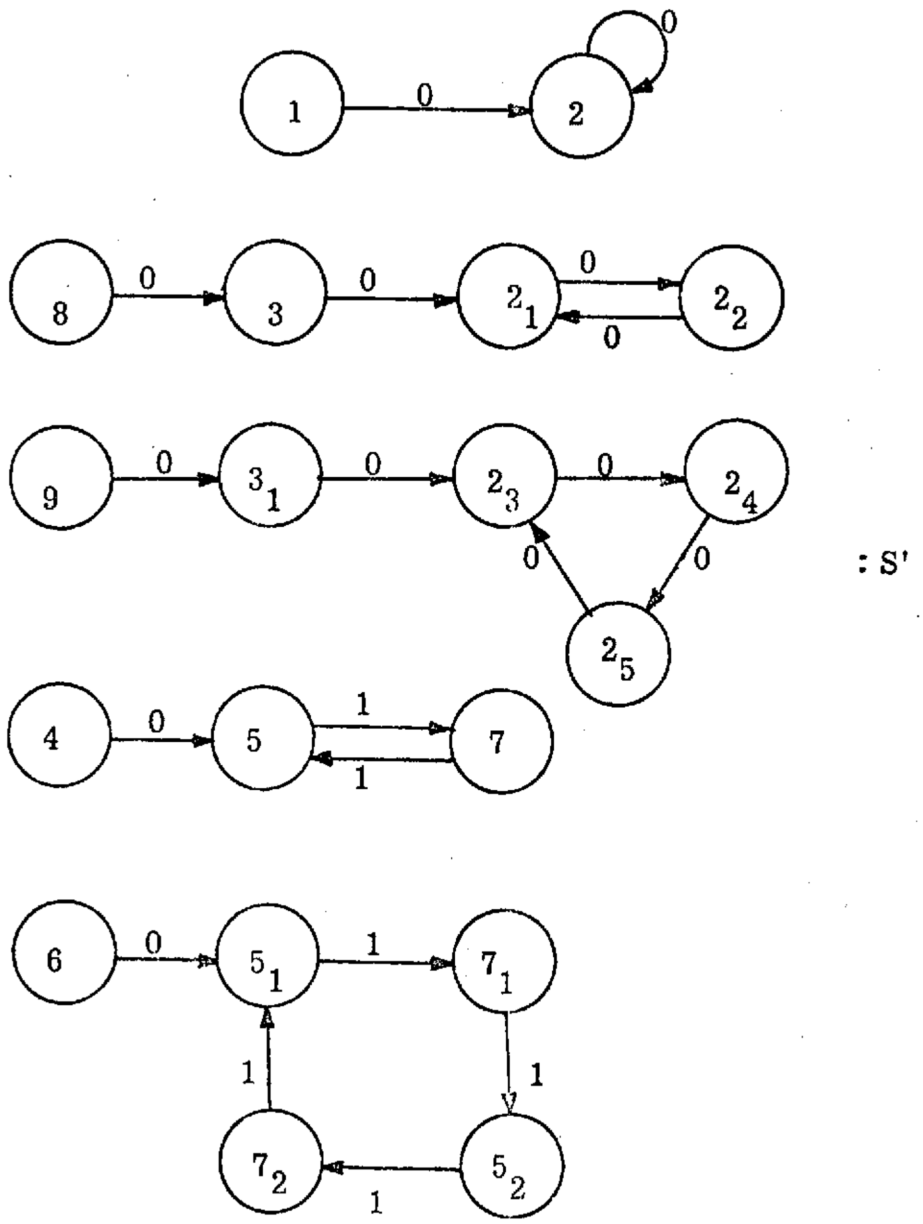
ii) Get $S = \{G_1, G_8, G_9, G_4, G_6\}$ from D_0, and renaming states such that, $\forall\, q \in Q$, all q's in S have the names $q, q^1, q^2, \ldots$ etc.



For $i \in Q$, $E'(i) = \{i\}$, except the following

$E'(2) = \{2, 2^1, 2^2\}$, $E'(3) = \{3, 3^1\}$, $E'(5) = \{5, 5^1\}$, $E(7) = \{7, 7^1\}$

iii) Construct S' by enlarging cycles in S so that no similar cycles exists; rename states such that, $\forall\, q \in Q$, all q's and q^i's have the names $q, q_1, q_2, \ldots$, etc.



$E(i) = \{i\}\ \forall\ i \in Q$ except the following

$E(2) = \{2, 2_1, 2_2, 2_3, 2_4, 2_5\}$

$E(3) = \{3, 3_1\}$

$E(5) = \{5, 5_1, 5_2\}$

$E(7) = \{7, 7_1, 7_2\}$

iv) On each cycle of S', change the output either of the arc which terminates on the root, or of any one arc if there is no root in that cycle.

$$\begin{tikzcd}
1 \arrow[r, "0"] & 2 \arrow[loop right, "0'"]
\end{tikzcd}$$

$$\begin{tikzcd}
8 \arrow[r, "0"] & 3 \arrow[r, "0"] & 2_1 \arrow[r, bend left, "0"] & 2_2 \arrow[l, bend left, "0'"]
\end{tikzcd}$$

$$\begin{tikzcd}
9 \arrow[r, "0"] & 3_1 \arrow[r, "0"] & 2_3 \arrow[r, "0"] & 2_4 \arrow[dl, "0"] \\
 & & 2_5 \arrow[u, "0'"] &
\end{tikzcd}$$

: S'

$$\begin{tikzcd}
4 \arrow[r, "0"] & 5 \arrow[r, bend left, "1"] & 7 \arrow[l, bend left, "1'"]
\end{tikzcd}$$

$$\begin{tikzcd}
6 \arrow[r, "0"] & 5_1 \arrow[r, "1"] & 7_1 \arrow[d, "1"] \\
 & 7_2 \arrow[u, "1'"] & 5_2 \arrow[l, "1"]
\end{tikzcd}$$

v) Embedding process.

	δ'/λ'	
Q' \ I	0	1
1	2/0	9/0
2	2/0'	1/1
2_1	$2_2/0$	1/1
2_2	$2_1/0'$	1/1
2_3	$2_4/0$	1/1
2_4	$2_5/0$	1/1
2_5	$2_3/0'$	1/1
3	$2_1/0$	4/1
3_1	$2_3/0$	4/1
4	5/0	1/1
5	7/1	8/0
5_1	$7_1/1$	8/0
5_2	$7_2/1$	8/0
6	$5_1/1$	1/1
7	5/1'	6/0
7_1	$5_2/1$	6/0
7_2	$5_1/1'$	6/0
8	3/0	9/0
9	$3_1/0$	8/0

$M'|0$ is clearly reduced. It is easy to verify that M' is a state-output augmentation of M.

To see that the size of the output set Z' of the state-output augmentation with RDS in Theorem 3.7 can not be reduced for machines in general, let M be a machine such that, $\forall$ a ϵ I and $\forall$ b ϵ Z, D_a contains two cycles with period 1 and output b. Then, in order to eliminate equivalent cycles in D_a, any state-output augmentation of M, say under $(\sigma_1, \sigma_2, \sigma_3)$, must have at least two distinct outputs b, b' such that $\sigma_3(b) = \sigma_3(b') = b$, $\forall$ b ϵ Z. Therefore the output set of the state-output augmentation must be at least twice as large as that of M.

In circuit terms, Theorem 3.7 says that, for any machine M that has no RDS, there is a state-output augmentation M' with RDS such that M' has one more output terminal than M. We will present one such augmentation M' which does not require any input and output translations.

Let M' be the state-output augmentation with RDS as constructed in the proof of Theorem 3.7. Let ℓ binary output terminals be used for M. If, for each $b_i \in Z$, b_i is assigned the value $(a_{i1}, a_{i2}, \ldots, a_{i\ell})$, where $a_{ij} \in \{0, 1\}$ is the value on the j^{th} output terminal, then let M' have $\ell + 1$ output terminals and b_i (and b_i') be assigned the value $(a_{i1}, a_{i2}, \ldots, a_{i\ell}, 0)$ (and $(a_{i1}, a_{i2}, \ldots, a_{i\ell}, 1)$) as shown schematically in Figure 3.4. During normal operation, M' simulates M using only the first ℓ output terminals. But when the machine M' is to be diagnosed, the outputs of all its $\ell + 1$ terminals will be observed so that the machine will have a RDS. Thus a diagnostic sequence employing this RDS can be applied to M'.

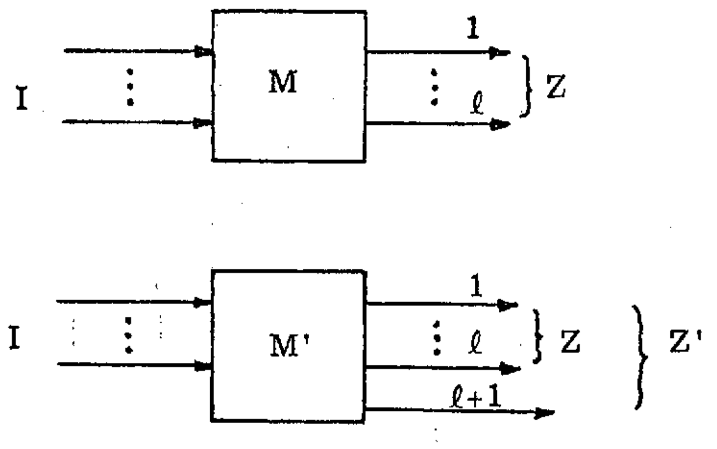
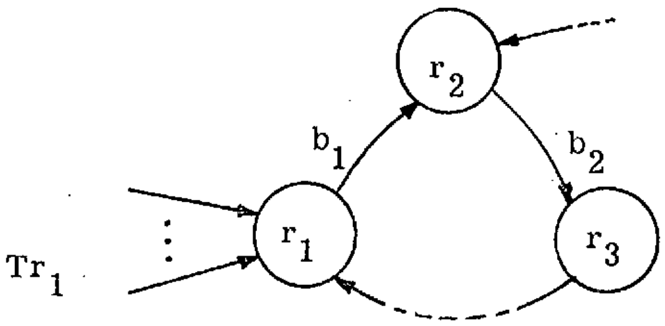


Figure 3.4 A Double-Z Augmentation Using neither Input nor Output Translations

It will be shown that for some machines, there exists state-output augmentations (with RDS) which have less outputs than that of double-Z augmentations.

A cycle C of a flower G is said to be <u>1 - peeled at the state r_1</u> if the state r_1 is split along with the in-tree, if any, as shown in Figure 3.5. Clearly every state q in the peeled flower is equivalent to the state q in the unpeeled G and $r_1' \equiv r_1$. In general, C is <u>k-peeled at the state r_1</u> if C is 1-peeled at the state r_1, and then k-1 peeled at the state r_2. It is obvious that if C is k-peeled (at any node of C), k $\geq$ (the period of C) - 1, then the flower containing C will have only one in-tree.

Let Z be the set of outputs of D_a. A <u>cover S for cycles</u> in D_a is a subset of Z such that for each cycle C in D_a there is a b ϵ S such that b is an output on an arc of C. Clearly $|S| \leq |Z|$.



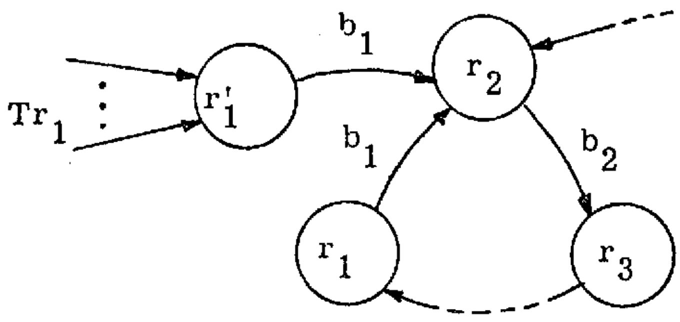


Figure 3.5 The Cycle C Is 1-peeled at r_1

Corollary 3.7.1

If M is a machine then there exists a state-output augmentation M' with RDS in $a \in I$ such that $|Z'| = |S| \cup |Z|$ where S is a minimum cover for cycles in D_a.

Proof:

The proof of this theorem is identical to that of Theorem 3.7 except in step ii) the cycle in each G_q of S is i-peeled at the root, for some integer i, such that the last arc peeled from the cycle carries an output in S.

Although the method suggested by Corollary 3.7.1 uses less output symbols, it usually needs more additional states than double-Z augmentations. Furthermore, adding outputs in augmentation, regardless of how few they are, often means that at least one extra output terminal will be needed. Since double-Z augmentations can also be implemented with one more output terminal, we will stick to double-Z augmentations in the following studies on how to minimize the number of states in augmentation with RDS.

Section 3. 5 Minimization of the State Set in a Double-Z Augmentation

To facilitate the process, we will consider a slightly restricted double-Z augmentation: M' is a double-Z augmentation for M under $(\sigma_1, \sigma_2, \sigma_3)$ such that $\forall\, b \in Z$, $\exists b, b' \in Z'$ for which $\sigma_3(b) = \sigma_3(b') = b$. Hence each symbol in Z corresponds to two symbols in Z'. We will try to minimize the number of states in double-Z augmentations by minimizing the additional states required to resolve convergences and to resolve equivalent and/or irreduced cycles in two separate processes. Since convergences can only occur at the in-tree part of a flower, we need only to consider in-trees in minimizing the number of states to resolve convergences.

3.5.1 Minimization of the State Set in Resolving Convergences

Let us consider first of all how a k-convergence in a flower G can be resolved (by double-Z augmentations) using the least number of additional states. Let S be a k-convergence at r_1 in G as shown in Figure 3.6, where only the convergence and the ensuing path of length $i + 1$, where $i = \lfloor \log_r k \rfloor$ ($\lfloor x \rfloor$ is the largest integer $\geq x$), are shown. Let us assume that all r_j's are states in an in-tree of G. The number k can be expressed as

$$k = a_o 2^i + a_1 2^{i-1} + \dots + a_{i-1} 2^1 + a_i 2^0 \qquad (3.1)$$

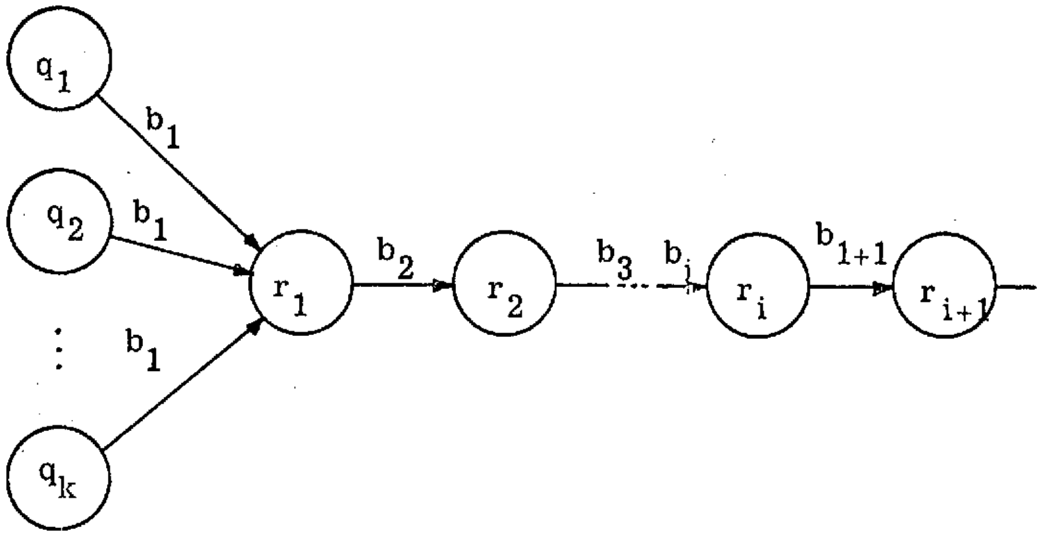


Figure 3.6. A k-Convergence and the Ensuing Path

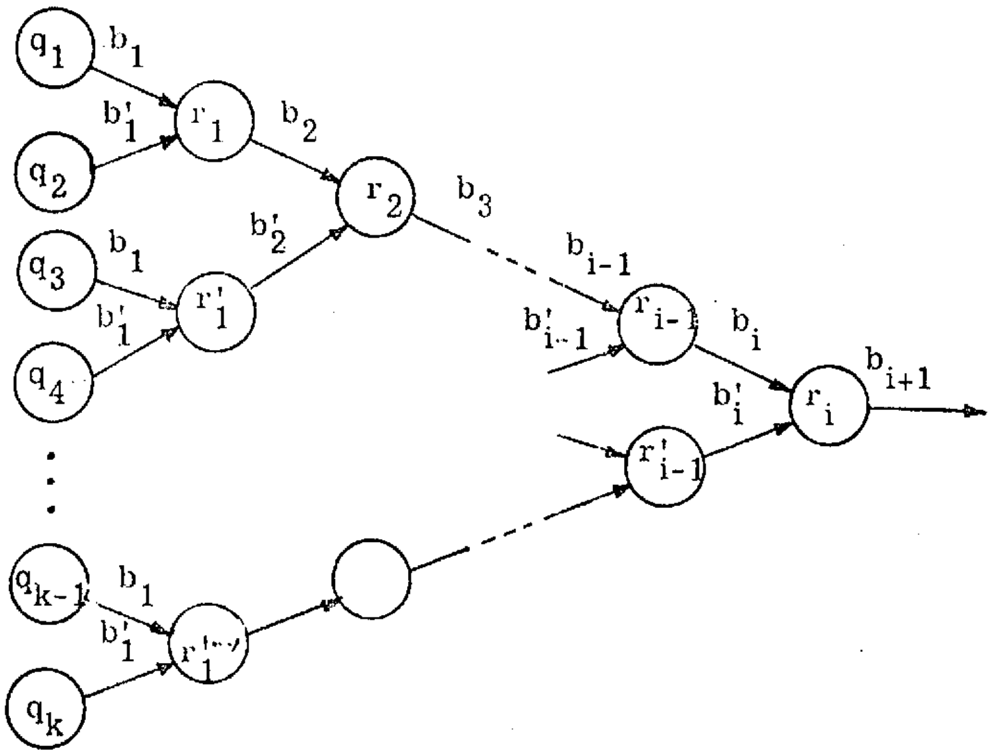


Figure 3.7 Resolve k-Convergence When k Is a Power of 2

where $a_j \in \{0, 1\}$, $\forall\ 1 \leq j \leq i$, and $a_0 = 1$. If k is a power of 2, then a k-convergence can be resolved using a double-Z augmentation as shown in Figure 3.7, where the in-tree containing this convergence is expanded.

The total number of states in Figure 3.6 can be easily counted to be 2k-1. On the other hand, if k is not a power of two, let $\{j_1, j_2, \ldots, j_s\}$ be the set of integers such that a_{j_t}, $1 \leq t \leq s$, in (3.1) equals 1, i.e.

$$k = \sum_{t=1}^{s} 2^{j_t} . \tag{3.2}$$

Then, for each j_t, the 2^{j_t} states in the k-convergence is a 2^{j_t}-convergence, which can be resolved as illustrated in Figure 3.7. Thus the k-convergence can be resolved as shown in Figure 3.8.

The total number of states in Figure 3.8 is therefore equal to k', where

$$k' \leq \sum_{t=1}^{s} (2 \cdot 2^{j_t}) + (j_1 - j_s - 1) \tag{3.3}$$

Since $j_1 = i = \lfloor \log_2 k \rfloor$, $j_s \geq 0$, and (3.2), we obtain

$$k' \leq 2k + \lfloor \log_2 k \rfloor - 1 \tag{3.4}$$

We will later refer to the method in Figure 3.8 as the <u>expanded in-tree method.</u>

Convergences with different outputs at the same state can be resolved by the same expanded in-tree as demonstrated in Figure 3.9.

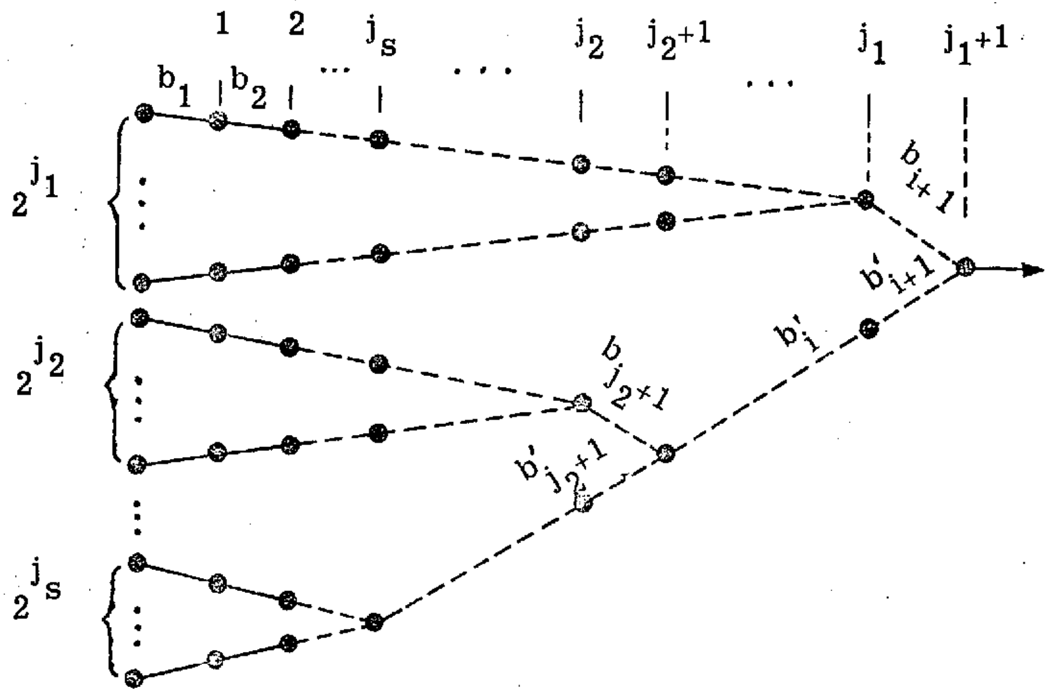


Figure 3.8 Resolve a k-Convergence Where k Is Not a Power of 2

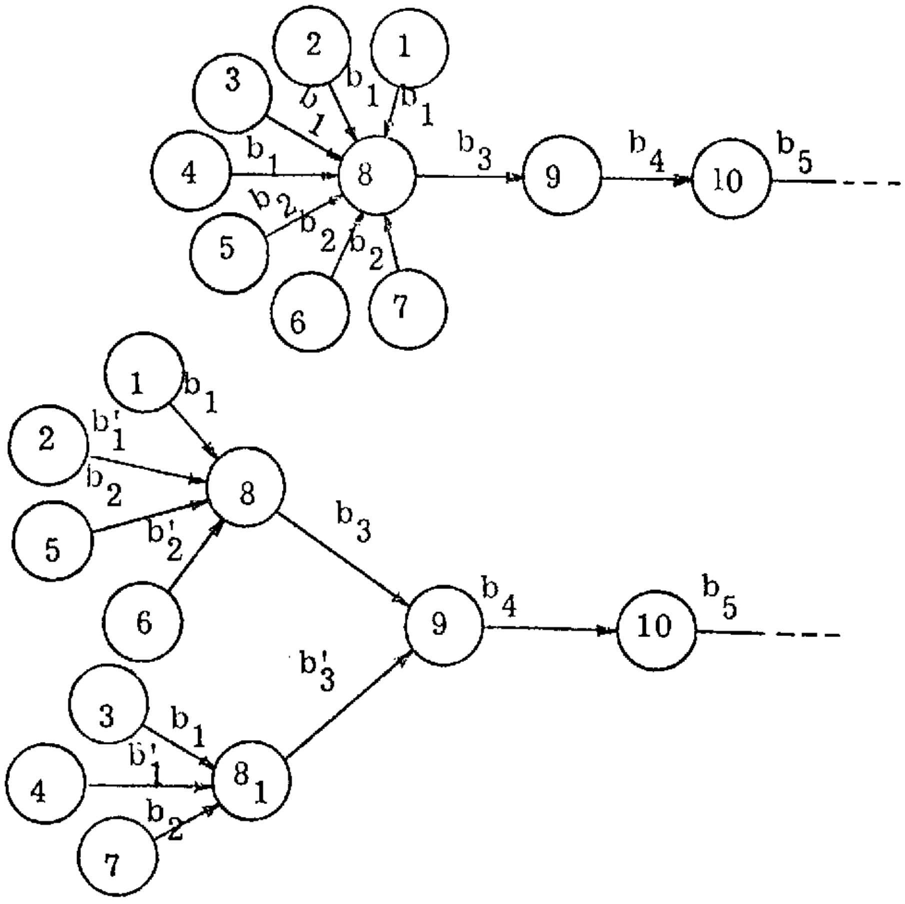


Figure 3.9 Resolving Two Convergences at the Same Node

Therefore a smaller convergence can always be resolved by the expanded in-tree which resolves a larger convergence that occurs at the same state.

It should be noted that, as in Figure 3.8, whenever a k-convergence at a state r_1 is resolved using the expanded in-tree method, then any convergence at r_j, $\forall\ 2 \leq j \leq i{+}1$, will get an additional incoming arc with the output b_j'. For example, in Figure 3.10, if the 3-convergence $\{1, 2, 3\}$ at the state 5 is resolved as shown, then the state 8 gets a new incident arc from the state 5, which has the output b_3'. This implies that the 2-convergence $\{5, 6\}$ at the state 8 can no longer be resolved simply by assigning a new output b_3' to the arc from the state 6 to the state 8. In fact, we have to resolve the convergence at the state 8 as a 3-convergence $\{5, 5_1, 6\}$. This type of added incoming arcs to a node as a result of resolving a convergence at a preceding node can create new convergences as well as enlarge convergences at some successive nodes. Resolving these new or enlarged convergences may again turn up other new convergences. This process may go on for a few iterations before it stops. As a result of such phenomena, this method for resolving convergence does not always employ a minimum number of states.

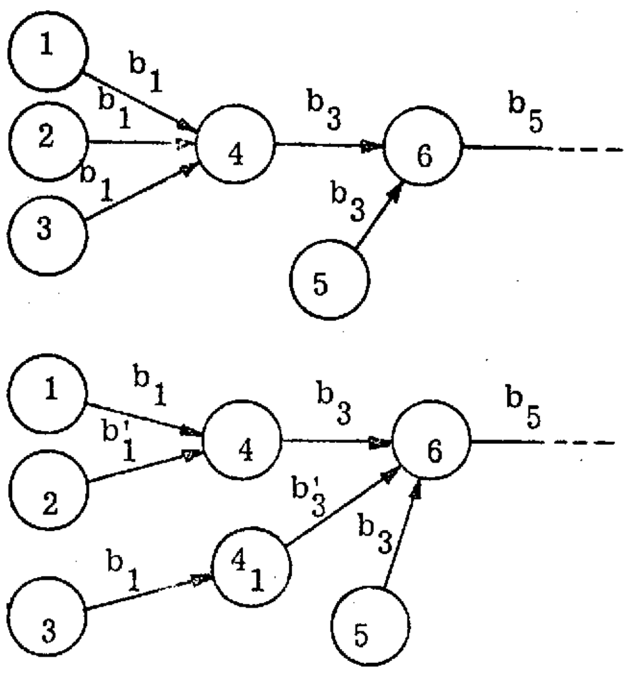


Figure 3. 10 An Enlarged Convergence As a Result of Resolving a Preceeding Convergence

Accordingly, convergences in a flower should be resolved starting from those which are farthest from the root and then those next farther and so on. However, convergences of the same distance from the root can be resolved in any order. Furthermore, convergences in a set of flowers, such as D_a of a machine, can be resolved by considering each flower independently. Although the expanded in-tree method provides us a way to minimize the number of states in resolving convergences, a bound on the number of states required in general should help us to decide whether this method is feasible in a given situation. Since using the expanded in-tree method may generate new convergences in a few iterations, which causes some complexities in deriving a better bound, a different strategy will be used.

3. 5. 2 An Upper Bound on the Number of States for Resolving Convergences

Let T be an in-tree in the flower G with the node t as its root. For any node r in T, let P_r denote the path from r to t. If S is a convergence at r, then S is separated at r if the branch in-tree at r together with a duplicated path P'_r are separated (except the root) from T as illustrated in Figure 3. 11. Note that a new convergence will be created at the root after a convergence is separated. Let s be a node in T such that s has an arc to the root t. Then the branch in-tree T_s at s is said to be 1-elongated if a path, which is equivalent to the path of the cycle C of G starting from t with a length equal to the period of C, is inserted between s and t as shown in Figure 3. 12. The operation "k-elongated" can be defined similarly as the 1-elongation except the inserted path between the branch in-tree and the root is a consecutive connections of k copies of the cycle.

We would like to point out here that all the operations we defined so far, namely k-peelings, in-tree expansions, convergence separations, and branch in-tree k-elongations are all state splitting operations. Hence, for a state q, any newly created states which are split from q as a result of these operations will be taken as a member of the set E(q) (see the proof of Theorem 3. 8 for the meaning and use of E(q)), and the embedding process can always be performed properly to produce a state-output augmentation with RDS. Hereafter we will merely describe how a reduced autonomous machine can be obtained with these operations and proper outputs assignments without further discussions on embedding procedures.

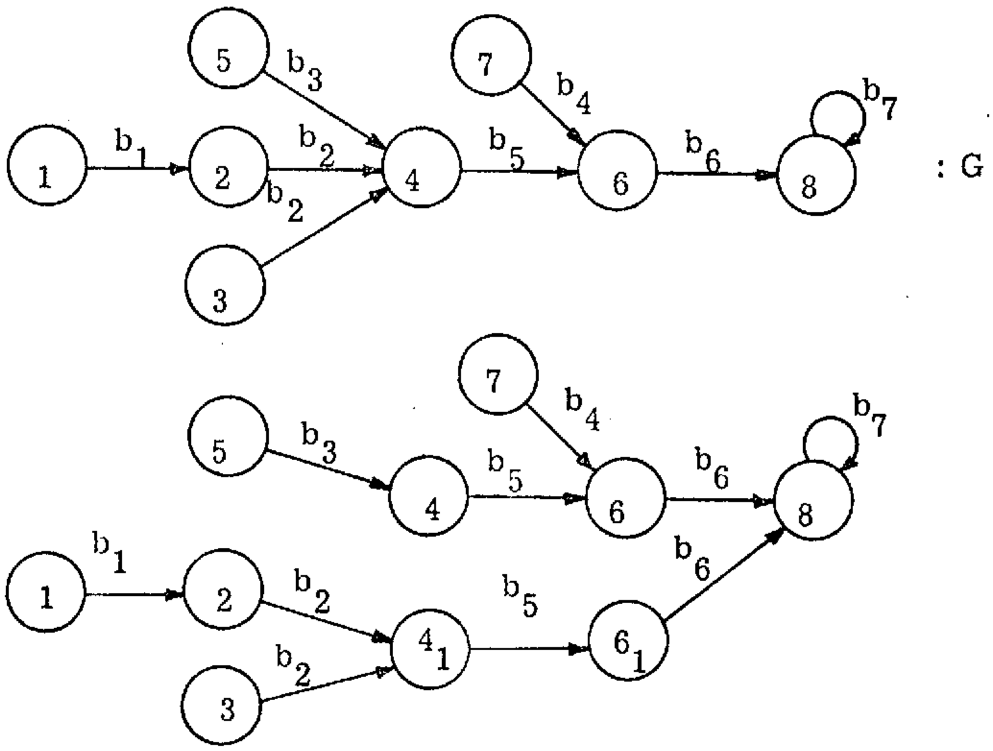


Figure 3.11 The Convergence $\{2, 3\}$ in G Is Separated at 4

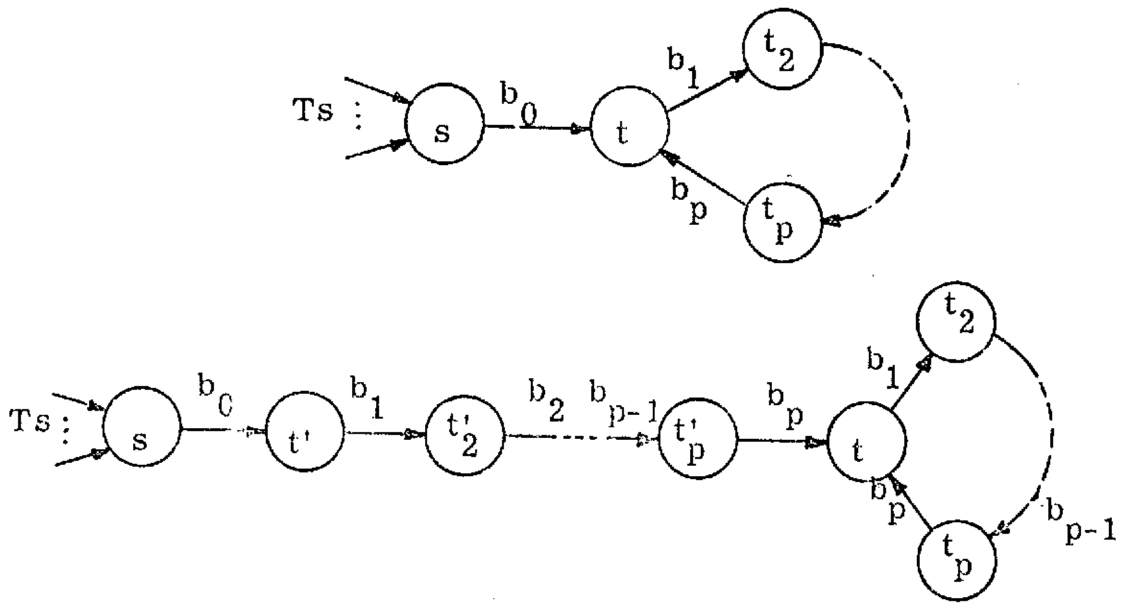


Figure 3.12 T_s Is 1-Elongated

Let C be the cycle of a flower G and p be the period of C. Let $\{S_1, S_2, \ldots, S_j\}$ be the set of all convergences in the in-trees of G; let q_i denote the state to which S_i converges. To obtain a bound on the number of states, the new strategy to resolve convergences in G is described in the following steps:

1) C is k-peeled, for some k, at any node of C so that G contains only one in-tree. Clearly $k \leq p-1$. Let t be the root.

2) Each convergence S_i is separated at q_i in the following order. That S_i which is farthest from the root t is separated first (convergences which are of the same distance may be separated in any order), then a convergence that is next farthest is separated, and so on. Consequently, in the resulting flower each branch in-tree T_{s_i} at s_i, where s_i has an arc to the root t, contains at most one convergence. Let this convergence be denoted as S_i', and let q_i' be the state at which S_i' converges. As a result of separations of j convergences in T_t, there is a j-convergence at the root t.

3) For each T_{s_i}, if $d_{q_i'}$, i.e. the distance from q_i' to the root t, is smaller than $\lfloor \log_2 |S_i'| \rfloor + 2$ then T_{s_i} is ℓ-elongated, where ℓ is the smallest integer such that $\ell \geq (\lfloor \log_2 |S_i'| \rfloor + 2) - d_{q_i'}$.

4) The in-tree T_t is ℓ'-elongated, where ℓ' is the smallest integer such that $\ell' p \geq \lfloor \log_2 j \rfloor + 2$.

Let G' be the flower resulted from executing the above four steps on G. First note that $|S_i'| \leq |S_i|$. Since the distance from q_i' to the root t is larger than $\lfloor \log_2 |S_i'| \rfloor + 2$ as a result of step 3), and since there is only one convergence on T_{s_i}, we may apply the expanded in-tree method on all S_i'-convergences without creating any new convergences.

Similarly, the j-convergence at t can be resolved without creating any new convergences. Let the number of states for resolving all convergences be m'. Then, from (3.4),

$$m' \leq \sum_{i=1}^{j} [2|S_i'| + \lfloor \log_2 |S_i'| \rfloor - 1] + 2j + \lfloor \log_2 j \rfloor - 1. \qquad (3.5)$$

Let $m = \sum_{i=1}^{j} |S_i|$, i.e., the number of all states in the convergences in the in-trees of G. Since $\lfloor \log_2 x \rfloor < x - 1$, and $|S_i'| \leq |S_i|$, (3.5) can be reduced to

$$m' \leq 3m + j. \qquad (3.6)$$

After all the convergences are resolved, the total number of states in the in-tree of G', denoted n_1', should include the number of all the other unaccounted states which is no more than $(n_1 - m) + \sum_{i=1}^{j} (d_{q_i} + p) + (p - 1)$ where n_1 is the number of states in the in-tree of G. Hence $(n_1 - m)$ is the number of states in the in-tree of G that is not in any convergences; $(d_{q_i} + p)$ is larger than the number of extra states in T_{s_i} (as the results of Step 1) and 3) which have not yet been counted in (3.6). Similarly the last term (p - 1) is larger than the number of states uncounted (as a result of Step 4). Hence

$$n_1' \leq 3m + j + (n_1 - m) + \sum_{i=1}^{j} (d_{q_i} + p) + (p - 1). \tag{3.7}$$

Clearly $d_{q_i} \leq (n_1 - m)$, hence (3.7) becomes

$$n_1' \leq n_1 + 2m + j(n_1 - m + p + 1) + (p - 1). \tag{3.8}$$

Let n denote the number of states in the flower G. Then clearly $n = n_1 + p - 1$. Since each convergence involves at least two states, $j \leq m/2$. Hence, from (3.8), we obtain

$$n_1' \leq n + m(n - m + 5)/2. \tag{3.9}$$

If a D_a instead of a single flower is considered, let n_t', n_ℓ, m_ℓ have the following meanings:

n_t':	the total number of states in the in-trees of D_a after resolving all convergences.
n_ℓ:	the number of states in the flower G_ℓ of D_a.
m_ℓ:	the number of states that are in a convergence in the in-trees of G_ℓ.

Then, by (3.9)

$$n_t' \leq \sum_{\ell} [n_\ell + m_\ell(n_\ell - m_\ell + 5)/2]. \tag{3.10}$$

Let n and m be, respectively, the number of states in D_a and the number of convergences in the in-trees of D_a. Then since $n = \sum_{\ell} n_\ell$ and $m_\ell \leq m$, (3.10) becomes

$$n_t' \leq n + m \sum_{\ell}{}' (n_\ell - m_\ell + 5)/2$$

$$= n + m(n - m + 5)/2 \,. \tag{3.11}$$

3.5.3 Minimization of the State Set in Resolving Cycles

As noted before, any number of equivalent or nonreduced cycles with different periods can be resolved by assigning a new output symbol to one arc of each cycle. Similar cycles can also be resolved with double-Z augmentation by assigning one of two output symbols to each arc of a cycle and by enlarging some cycles so that more cycles can be resolved.

After the convergences in a D_a are resolved, in some flower of D_a the outputs on the two arcs (one of the cycle and the other of the in-tree) which terminate on the root may have to be assigned different outputs, e.g. b_j and b_j', to resolve the convergence at the root. Hence the output on such an arc of a cycle should not be changed during output assignments in resolving cycles, otherwise a convergence at the root resolved before would reappear. In order to minimize the number of states in resolving cycles in a D_a, the output assignments should start with the class of similar cycles which has the smallest period. Assign outputs b_i or b_i' on each arc of cycles in this class if the output of that arc was b_i, so that as many nonequivalent and reduced cycles can be produced. If after doing this there are still some similar cycles left unresolved in this class, then enlarge these unresolved cycles once. Repeat the above procedure on a class of similar cycles with smallest period, and so on until all similar cycles are resolved.

3.5.4 An Upper Bound on the Number of States in Resolving Cycles

From our previous discussion, it is clear that the enlargement of some cycles may be necessary only during resolving similar cycles. Furthermore, less similar cycles can be resolved by a fixed output set if the period of those cycles is smaller. If the output on one arc of the cycle is not allowed to be changed, e.g., when it is fixed from resolving the convergence at the root, then the number of similar cycles that can be resolved is further reduced. Therefore the worst case for resolving cycles in a D_a is when all the cycles in D_a are similar with a period equal to one and, in each cycle of D_a, the output of the arc that terminates on the root is fixed. Let us assume that D_a is so. Thus if a cycle in D_a is enlarged p times then only the outputs on p - 1 arcs of the cycle may be assigned either one of two values. Let J_p denote the number of nonequivalent and reduced cycles with period p, where the outputs on p-1 arcs of each cycle have the values in, say, $\{b, b'\}$ and, for all cycles, the output on the remaining (one) arc is fixed to be, say, b. Let the number of cycles in D_a be n_c. Since all the cycles are similar with period 1, n_c is also the number of states in cycles of D_a. Let D'_a be the D_a with all its cycles resolved. Let n'_c be the number of states in the cycles of D'_a, and let ℓ be the largest period of cycles in D'_a. Then the smallest possible ℓ must satisfy the inequality,

$$\sum_{i=1}^{\ell-1} J_i \leq n_c \leq \sum_{i=1}^{\ell} J_i. \qquad (3.12)$$

Clearly then

$$n'_c = \sum_{i=1}^{\ell-1} (i \cdot J_i) + \ell \cdot (n_c - \sum_{i=1}^{\ell-1} i \cdot J_i) \qquad (3.13)$$

where $\sum_{i=1}^{\ell-1} (i \cdot J_i)$ is the number of reduced and nonequivalent cycles with period ℓ - 1 or less, and $\ell \cdot (n_c - \sum_{i=1}^{\ell-1} i \cdot J_i)$ is the number of reduced and nonequivalent cycles with period ℓ.

For small values of n_c's, the corresponding values for ℓ's and (n'_c)'s are shown in Table 3.1.

n_c	ℓ	n'_c
1	1	1
2	2	3
3	3	6
4	3	9
5	4	13
6	4	17
7	4	21
8	5	26
9	5	31
10	5	36
11	5	41
12	5	46
13	5	51
14	6	57

Table 3.1
Small Values of n_c, ℓ, And n'_c.

Recall the Möbius formula [17],

$$I_p(k) = \frac{1}{p} \sum_{d|p} \mu(d)\, k^{p/d} \tag{3.14}$$

where

$$\mu(d) = \begin{cases} 1 & \text{if } d = 1 \\ (-1)^j & \text{if } d \text{ is the product of } j \text{ distinct primes} \\ 0 & \text{if } d \text{ contains any repeated prime factors.} \end{cases}$$

$I_p(k)$ is the number of all irreducible polynomials with degree p over a finite field of k elements. Each irreducible polynomial of degree p over a finite field of k elements corresponds uniquely to a cyclic generator, and the sequence generated by each such cyclic generator is just the output sequence of a reduced cycle with period p and with the output of each arc one of k values. Therefore $I_p(2)$ is equal to the number of reduced and nonequivalent cycles with period p and with the output on each arc one of two values. Even though the output on one of the arcs in each cycle is fixed in our case, obviously the output sequences of at least one half of the cycles counted by $I_p(2)$ can be used as output assignments to a cycle in D_a with period p. Thus

$$J_p \geq \frac{1}{2} I_p(2)\,. \tag{3.15}$$

We will use I_p as an abbreviation for $I_p(2)$.

If ℓ' is the integer such that

$$\frac{1}{2} \sum_{i=1}^{\ell'-1} I_i \leq n_c \leq \frac{1}{2} \sum_{i=1}^{\ell'} I_i \tag{3.16}$$

Then, because $J_i \geq \frac{1}{2} I_i$, comparing (3. 12) and (3. 16), we have

$$\ell' \geq \ell . \tag{3.17}$$

From (3. 14), I_i can be expressed as

$$I_i = \frac{1}{i} (2^i - \sum_{j=1} a_j 2^{b_j}), \tag{3.18}$$

where $a_j \in \{0, 1, -1\}$, $a_1 = 1$, $b_{j_1} > b_{j_2}$ if $j_1 > j_2$, and

$$\begin{cases} b_1 = i/2 & \text{if} \quad i \text{ is even} \\ b_1 \leq \lfloor i/3 \rfloor & \text{if} \quad i \text{ is odd.} \end{cases} \tag{3.19}$$

Let us define I_i' as

$$I_i' = \frac{1}{i} (2^i - 2^{b_1+1}) . \tag{3.20}$$

Since $2^s = \sum_{t=1}^{s-1} 2^t + 1$, comparing (3. 18) and (3. 20) and noting the value of b_1 as shown in (3. 19), it is clear that

$$I_i \geq I_i' . \tag{3.21}$$

It is seen that both I_i and I_i' are monotonically increasing functions of i. Thus let p be the smallest integer such that

$$\frac{1}{2} \sum_{i=1}^{p-1} I_i' \leq n_c \leq \frac{1}{2} \sum_{i=1}^{p} I_i'. \tag{3.22}$$

Then $p \geq \ell'$. From (3. 20) and (3. 22), we obtain

$$n_c \geq \frac{1}{2} \sum_{i=1}^{p-1} \frac{1}{i} (2^i - 2^{b_1+1})$$

$$\geq \frac{1}{2(p-1)} \sum_{i=1}^{p-1} (2^i - 2^{b_1+1})$$

$$\geq \frac{1}{2(p-1)} \left(2^{p} - 2^{\lfloor p/2 \rfloor + 1}\right)$$

$$\geq \frac{1}{2(p-1)} \; 2^{p-1} \; . \tag{3.23}$$

It can be shown that

$$2^{i}/i \;\geq i \qquad \forall\, i \geq 4,$$

hence from (3.23),

$$n_c \geq \frac{1}{2}\,(p-1) \qquad \forall\, p \geq 5.$$

Substituting this into (3.23), we obtain

$$n_c \geq \frac{1}{4n_c} \; 2^{p-1} .$$

Thus $n_c^2 \geq 2^{p-3}$. Take $\log_2$ on both sides then

$$2\log_2 n_c \;\geq (p-3)$$

or

$$p \leq 2\log_2 n_c + 3 \qquad \forall\, p \geq 5. \tag{3.24}$$

Recall that ℓ is the largest period of cycles in D_a' and $p \geq \ell' \geq \ell$, hence

$$n_c' \leq pn_c \; . \tag{3.25}$$

Substituting (3.24) into (3.25), we obtain an upper bound

$$n_c' \leq n_c(3 + 2\log_2 n_c) \qquad \forall\, \ell \geq 5. \tag{3.26}$$

By careful examination on Table 3.1, we found that (3.26) is also true for $\ell \leq 4$, hence

$$n_c' \leq n_c(3 + 2\log_2 n_c) . \tag{3.27}$$

3.5.5 An Upper Bound on the Number of States Required for Double-Z Augmentations

Combining (3.11) and (3.27), an upper bound on the number of states required for a double-Z augmentation, denoted n', for any n-state machine can be obtained as

$$n' \leq n + m(n - m + 5)/2 + n_c(3 + 2\log_2 n_c). \qquad (3.28)$$

Since m, i.e. the number of convergences, at worst can be of the order n, n' is, in general, of the order n^2. In circuit terms, it says that a double-Z augmentation needs at most double the number of memory elements of the original machine.

Section 3. 6 Summary

Various properties of state graphs have been extracted to establish, firstly, an upper bound on the length of an RDS, then a necessary and sufficient condition for a machine to possess an RDS. The latter has served as a base from which methods to construct state-output augmentations with RDS's have been devised.

Various augmentations of machines have been defined. Augmentations with RDS's have been viewed as a way to improve the diagnosability of a sequential machine. After an evaluation of advantages and disadvantages of various augmentations, input augmentations with RDS's and state-output augmentations with RDS's have been selected as candidates. Since the former have been studied by other researchers, our major effort has been devoted to investigating state-output agumentations with special attention to minimizing the size of the output set of an augmented machine. It has been found that as few as twice the number of outputs of the given machine is sufficient for constructing a state-output augmentation with RDS. This is called the double-Z augmentation. It has been demonstrated that there exists an efficient way to implement a double-Z augmentation which does not require any input/output translations to simulate the given machine. Techniques for minimizing the number of states in resolving convergences and in resolving equivalent and nonreduced cycles have been developed. An upper bound on the number of states required in each case has been derived, and the sum of them results in an upper bound on the number of states required for double-Z augmentations. This bound reveals that, in worst cases, we have to double the number of memory elements of a machine to obtain a double-Z augmentations.

REFERENCES

[1] E. F. Moore, "Gedanken-Experiments on Sequential Machines," in Automata Studies (Annuals of Mathematical Studies), Princeton, N. J.: Princeton University Press, 1956, pp. 129-153.

[2] F. C. Hennie, "Fault Detecting Experiments for Sequential Circuits," in Proc. 5th Annual Symposium on Switching Theory and Logical Design, Nov. 1964, pp. 95-110.

[3] C. R. Kime, "An Organization for Checking Experiments on Sequential Circuits," IEEE Trans. Electronic Comp. (Short Notes), Vol. EC-15, Feb. 1966, pp. 113-115.

[4] Z. Kohavi and P. Lavallee, "Design of Sequential Machines with Fault-Detection Capabilities," IEEE Trans. Electronic Comp., Vol. EC-16, August 1967, pp. 473-484.

[5] I. Kohavi and Z. Kohavi, "Variable-Length Distinguishing Sequences and Their Application to the Design of Fault-Detection Experiments," IEEE Trans. Comp. (Short Notes), Vol. C-17, August 1968, pp. 792-795.

[6] G. Gönenc, "A Method for the Design of Fault Detection Experiments," IEEE Trans. Comp., (Short Notes), Vol. C-19, June 1970, pp. 551-558.

[7] E. P. Hsieh, "Checking Experiments for Sequential Machines," IEEE Trans. Comp., Vol. EC-20, Oct. 1971, pp. 1152-1166.

[8] L. Hsieh, "Checking Experiments for Sequential Machines," Department of Electrical and Computer Engineering, Systems Engineering Lab., The University of Michigan, SEL Technical Report No. 65, June 1972.

[9] M. J. Y. Williams and J. B. Angell, "Enhancing Testability of Large-Scale Integrated Circuits Via Test Points and Additional Logic," IEEE Trans. Comp., Vol. C-22, January 1973, pp. 46-60.

[10] J. F. Meyer and K. Yeh, "Diagnosable Machine Realizations of Sequential Behavior," Digest of 1971 International Symposium on Fault-Tolerant Computing, March 1971, pp. 22-25.

[11] C. E. Holborow, "An Improved Bound on the Length of Checking Experiments for Sequential Machines with Counter Cycles," IEEE Trans. Comp., Vol. C-21, June 1972, pp. 597-598.

[12] M. A. Arbib, Theories of Abstract Automata, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1969.

[13] J. A. Brzozowski, "Derivatives of Regular Expressions," J. ACM, Vol. 11, Oct. 1964, pp. 481-494.

[14] J. F. Meyer and B. P. Zeigler, "On the Limit of Linearity," Theory of Machines and Computations, Academic Press, Inc., New York and London, 1971, pp. 229-242.

[15] R. J. Leake, "Realization of Sequential Machines," IEEE Trans. Comp., (Correspondence), Vol. C-17, Dec. 1968, p. 1177.

[16] J. Hartmanis and R. E. Stearns, Algebraic Structure Theory of Sequential Machines, Prentice Hall, Englewood Cliffs, New Jersey, 1966.

[17] E. R. Berlekamp, Algebraic Coding Theory, McGraw-Hill Book Co., New York, 1968.

[18] L. Hsieh, Ph. D. Dissertation, Computer, Information and Control Engineering Program, University of Michigan (In preparation).