INPUT/OUTPUT CONTROL OF ASYNCHRONOUS MACHINES WITH RACES

By

JUN PENG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2007

To Mom and Dad.

ACKNOWLEDGMENTS

I thank my advisor, Dr. Jacob Hammer, for his excellent, professional guidance and support during my four years at the University of Florida. I thank Dr. John Schueller, Dr. Haniph Latchman, and Dr. Janise McNair, for serving on my Ph.D. supervisory committee and for their continuous help and advice.

I thank my officemates, Debraj Chakraborty and Niranjan Venkatraman, for interesting and joyful discussions. I thank all my friends for their care and friendship.

Last but not least, I thank my brother and his wife for their understanding, support and love throughout my school years. I thank my parents, Mingqing Peng and Mingxia Wu, whose unceasing love and whole-hearted support made finishing this work possible. I thank my husband, Zhipeng Liu, for his love and faith in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

INPUT/OUTPUT CONTROL OF ASYNCHRONOUS MACHINES WITH RACES
By

Jun Peng

August 2007

Chair: Jacob Hammer
Major: Electrical and Computer Engineering

The occurrence of races causes unpredictable and undesirable behavior in asynchronous sequential machines. In the present work, traditional feedback control techniques are used to control a race-afflicted machine, so as to turn it into a deterministic machine that matches a desired model. Instead of replacing or redesigning the whole machine, I add an output feedback controller to the original defective machine, and the controller eliminates the negative effects of the critical races. The present work focuses on asynchronous sequential machines in which the state of the machine is not provided as an output.

The results include the necessary and sufficient conditions for the existence of controllers that eliminate the effects of a critical race, as well as algorithms for their design. The necessary and sufficient conditions for the existence of controllers are presented in terms of certain matrix inequalities.

CHAPTER 1
INTRODUCTION

Asynchronous sequential machines are digital logic circuits without synchronizing clock, so they are also called clockless logic circuits. The lack of a synchronizing clock allows asynchronous machines to operate much faster. In addition, there are practical applications, such as parallel computation, where the underlying system is inherently asynchronous. Asynchronous design techniques can also be used to achieve maximum efficiency in parallel computation (Cole and Zajicek (1990), Higham and Schenk (1992), Nishimura (1995)). The design of asynchronous machines has been an active area of research since mid 1950s (Huffman (1954, 1955)). Potential difficulties, such as critical races and infinite cycles that may arise in the design of asynchronous machines are discussed in the literature (Kohavi (1970), Unger (1959)).

Both critical races and infinite cycles are flaws in the operation of an asynchronous sequential machine. In this dissertation, I will focus on asynchronous machines with critical races. A critical race drives the machine to exhibit nonpredictable behavior, and it may be caused by malfunctions, by design flaws, or by implementation flaws. Common practice is to rebuild a machine that is afflicted by a critical race, and replace it with a race-free machine.

In the present work, I use traditional feedback control techniques to control a race-afflicted machine, so as to turn it into a deterministic machine that matches a desired model. Instead of replacing or redesigning a defective machine, I add an output feedback controller, and the controller eliminates the effects of the critical races. The feedback controller turns the closed loop system into a deterministic machine, and the closed loop system imitates the desired model (Figure 1-1).

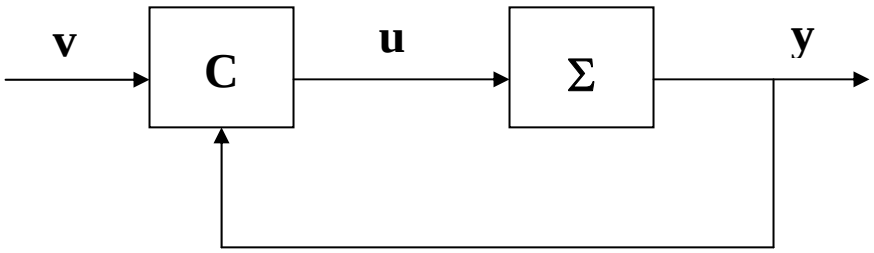

Figure 1-1. Control configuration for the asynchronous machine Σ

Here, Σ is the machine being controlled and C is another asynchronous machine that serves as an output feedback controller. We denote by Σ_c the machine described by the closed loop. The objective is to find a controller C for which the closed loop machine Σ_c exhibits desirable behavior.

I represent the behavior desired for the closed loop system by an asynchronous machine Σ', called a model. In these terms, the objective is to design a controller C for which the closed loop machine Σ_c simulates the model Σ'. Of course, the machine Σ' that represents the desired behavior is not afflicted by any critical races. Thus, by simulating the behavior of Σ', the closed loop system eliminates the ill effects of the critical races present in Σ. The problem of designing such a controller C is often referred to as the model-matching problem. The objective, then, is to find necessary and sufficient conditions for the existence of the controller C that solves the model matching problem. When such a controller exists, I also provide an algorithm for its construction.

The literatures regarding the model-matching problem of asynchronous machines with races has focused so far on asynchronous sequential machines in which the state is provided as the output of the machines (input/state machines). In Murphy *et al.* (2002, 2003), the control of asynchronous machines was discussed and state feedback controllers that eliminate the effects of critical races in asynchronous machines were developed. In Venkatraman and Hammer (2004),

state feedback controllers were used to eliminate the effects of infinite cycles on asynchronous machines; and in Geng and Hammer (2005), the problem of model matching with output feedback controllers was considered for asynchronous machines with no critical races. In the present work, I concentrate on the problem of designing output feedback controllers that eliminate the effects of critical races in asynchronous machines. The problem of eliminating the effects of critical races with output feedback requires the development of additional notions as well as the development of new design algorithms for controllers, and these are the subjects of our present discussion. To introduce these notions, I start with a brief review of some of the underpinnings of the theory of asynchronous machines.

Unlike a synchronous machine, which is driven by clock pulses, an asynchronous machine is driven by changes of its input variables. A stable state is a state at which the machine lingers until a change occurs in one of its input variables. In general, the change of an input variable causes an asynchronous machine to go through a succession of state transitions. If this succession of transitions ends, then the final state reached by the machine is a stable state; the states through which the machine passes during the succession are unstable states. Ideally, an asynchronous machine passes through an unstable state in zero time. Thus, unstable states are not noticeable to the user.

If an asynchronous machine has a succession of state transitions that does not terminate, then the machine has an infinite cycle. Infinite cycles form another class of potential defects of an asynchronous machine. The elimination of the effects of infinite cycles by the use of state feedback was discussed in Venkatraman and Hammer (2004). Asynchronous machines with infinite cycles are not discussed in this dissertation; I assume that all machines under consideration have no infinite cycles.

To guaranty the proper behavior of an asynchronous machine, some care has to be exercised during its operation. In particular, one has to avoid changing values of input variables while the machine undergoes a succession of state transitions. If an input change occurs while the machine is not in a stable state, then, due to asynchrony, it is not possible to predict the state of the machine at the instant in which the input change occurs. As the response of the machine depends on its state, this may result in an unpredictable response of the machine. In other words, the response may vary depending on the specific state of the machine at the instant of the input change. To avoid this situation, asynchronous machines are normally operated so as to guaranty that input changes occur only while the machine is in a stable state. When this precaution is taken, we say that the machine operates in fundamental mode. In this dissertation, all asynchronous machines are operated in fundamental mode.

The development of necessary and sufficient conditions for the existence of a model matching controller C and the algorithm for its construction depend on a certain generalized concept of state, introduced in chapter 2 below. A generalized state describes a persistent state of the machine Σ about which only partial information is available. More specifically, as Σ is an input/output machine, it is not always possible to determine its current state from available input/output data. A generalized state indicates a situation in which it is known that Σ is in a stable state, but the exact state of Σ is not known; the machine can be in any one of a pre-determined set of stable states. The generalized state allows us to use the partial information available about the state of Σ to continue controlling the machine as best as possible toward the goal of achieving model matching, while taking best advantage of the available information about Σ. The generalized state allows us to formalize in a concise and functional way the future implications of uncertainties in the present state of the machine Σ.

The notion of a generalized state was also used in Venkatraman and Hammer (2004) to represent phenomena related to the presence of infinite cycles. In the present paper, I show that a generalized notion of state can also be used to represent uncertainty in asynchronous machines with critical races, in situations where the exact state of the machine is not known.

The mathematical background of our discussion is based on Eilenberg (1974). Studies dealing with other aspects of the control of sequential machines can be found in Ramadge and Wonham (1987) and in Thistle and Wonham (1994), where the theory of discrete event systems is investigated; in Ozveren, Willsky, and Antsklis (1991), where stability issues of sequential machines are analyzed; and in Hammer (1994, 1995, 1996a and b, 1997), Dibenedetto, Saldanha, and Sangiovanni-Vincentelli (1994), Barrett and Lafortune (1998), where issues related to control and model matching for sequential machines are considered. These discussions do not take into consideration specialized issues related to the function of asynchronous machines, like the issues of stable states, unstable states, and fundament mode operation. As a result, these works refer mostly to the control of synchronous machines.

CHAPTER 2
TERMINOLOGY AND BACKGROUND

2.1 Asynchronous Sequential Machines

Definition 2-1 An asynchronous sequential machine Σ is defined by a sextuple (A, Y, X, x_0, f, h), where A, Y and X are nonempty finite sets: A is the input set, Y is the output set, and X is the state set. $x_0 \in X$ is the initial state of the machine. The partial function $f : A \times X \rightarrow X$ is the state transition function (or recursion function) and the partial function $h : A \times X \rightarrow Y$ is the output function. When the output function h does not depend on the input character (i.e., when $h : X \rightarrow Y$), the machine Σ is called a Moore machine in Moore (1956). ♦

Note that every asynchronous machine can be represented as a Moore machine. The machine Σ operates according to a recursion form

$$\begin{aligned} x_{k+1} &= f(x_k, u_k), \\ y_k &= h(x_k),\ k = 0, 1, 2, \ldots \end{aligned} \tag{2-1}$$

Where, k counts the steps of the machine Σ. The sequences x_k, u_k, and y_k are the state sequence, the input sequence and the output sequence, respectively. ♦

The machine Σ is an input/state machine if $Y = X$, or $y_k = x_k$ for each step $k \geq 0$. When the output is not the state, then the machine is an input/output machine. The present paper focuses on input/output machines.

Definition 2-2 Let Σ be an asynchronous machine represented by sextuple (A, Y, X, x_0, f, h). A pair $(x, u) \in X \times A$ is called a valid pair if the recursion function f is defined at it. If $x = f(x, u)$, then the combination (x, u) is a stable combination. ♦

Definition 2-3 Let (x, u) be a valid pair of the machine $\Sigma = (A, Y, X, x_0, f, h)$. If (x, u) is not a stable combination, then the machine generates a chain of transitions $x_1 = f(x, u)$, $x_2 = f(x_1,$

u), …. If the chain does not terminate, then the machine Σ contains an infinite cycle. If the succession of transitions ends at a stable combination (x_i, u), then x_i is the next stable state of x with the input character u. ♦

In the present work, I assume that none of our asynchronous machines possess infinite cycles.

Definition 2-4 Let Y be an alphabet and let $y_1, \ldots, y_q \in Y$ be a list of characters such that $y_{i+1} \neq y_i$ for all $i = 1, \ldots, q - 1$. Then, the burst of a string

$$y = y_1 y_1 \cdots y_1 y_2 y_2 \cdots y_2 \cdots y_q y_q \cdots y_q \text{ is } \beta(y) := y_1 y_2 \cdots y_{q-1} y_q$$

and $\beta_{-1}(y) := y_1 y_2 \ldots y_{q-1}$, for $q > 1$; $\beta_{-1} := \varnothing$, for $q = 1$.

Let $x_1 x_2 x_3 \ldots x_m$ be the string of states generated by the machine Σ from valid pair (x, u) and x_m is the next stable state. Then, the burst of the valid pair (x, u) is defined as

$$\beta(x_m, x, u) := \beta(h(x)h(x_1)h(x_2)\ldots h(x_{m-1})h(x_m)). \quad ♦$$

Definition 2-5 Let p_1 and p_2 be two strings of the alphabet A. As usual, p_2 is a prefix of p_1 if there is a string p_3 such that $p_1 = p_2 p_3$. We say that p_2 is a strict prefix of p_1 if $p_3 \neq \varnothing$, the empty string.♦

For example, given three strings p_1, p_2, and p_3: $p_1 = y_1 y_1 y_2 y_3$; $p_2 = y_1 y_1 y_2 y_3$; and $p_3 = y_1 y_1 y_2 y_3 y_2 y_3$. The string p_1 and p_2 are each other's prefix string. Both p_1 and p_2 are strict prefix strings of the string p_3.

Definition 2-6 A state-input pair (r, v) for which the next stable state of the machine is unpredictable is called a critical race pair, or, briefly, a critical race.

There may be more than one possible next stable states of the critical race pair (r, v). Let us suppose there are m possible next stable states of (r, v). The set of all these states $\{r^1, r^2, \ldots,$

$r^m\}$ is called the outcomes of the race. Correspondingly, there are m bursts for the critical race pair (r, v), one for each possible outcome of the race. Let β_i be the burst generated by the machine Σ when the outcome of the race is r^i. Then, we refer to the set $\beta(r, v) := \{\beta(r^1, r, v), \beta(r^2, r, v), \ldots, \beta(r^m, r, v)\}$ as the burst set of the critical race (r, v).♦

More details about races of asynchronous machines can be found in Kohavi (1970).

Definition 2-7 For a deterministic asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$, let x' be the next stable state of a valid pair (x, u). The stable recursion function $s : X \times A \rightarrow X$ of Σ is given by $s(x, u) := x'$ for all valid pairs $(x, u) \in X \times A$. The stable state machine induced by Σ is represented by the sextuple $\Sigma_{|s} = (A, Y, X, x_0, s, h)$.

For an asynchronous machine Σ with a critical race pair (r, v), the stable recursion function s has multiple values at the pair (r, v), say, $s(r, v) := \{r^1, r^2, \ldots, r^m\}$. Here, $r^1, r^2, \ldots, r^m$ are the outcomes of the critical race (r, v). ♦

Definition 2-8 Let $X^r := \{x^{i(1)}, ..., x^{i(m)}\}$ be a set of states of the machine Σ, and assume they have a non-empty set U of common input characters. For an element $u \in U$, let $s[X^r, u]$ be the set of all possible next stable states, where s is the stable transition function of the machine Σ. Let $B(X^r,u)$ be the set of all bursts from X^r to $s[X^r, u]$. For each burst $\beta \in B(X^r, u)$, let $X(\beta) \subset s[X^r,u]$ be the set of all states $x \in s[X^r, u]$, i.e., the set of all states that can be reached from X^r via the burst β. We refer to $X(\beta)$ as a burst equivalent set for the burst β with respect to input u.♦

Note that the burst equivalent sets in $s[X^r, u]$ may not be disjoint.

The following is an example to show how to obtain the stable state machine and transition equivalent set for a given subset of its state set. Consider a machine Σ with the input alphabet

A={a, b, c}, the output alphabet Y={0, 1, 2}, and the state set X={x^1, x^2, x^3, x^4 }. There is a critical race pair (x^1, c) in the machine. This machine Σ is depicted by a chart (Table 2-1) or a figure (Figure 2-1). So is the stable state machine $\Sigma_{|s}$ of Σ (Table 2-2, Figure 2-2).

Table 2-1. Transition table of the machine Σ

	a	b	c	Y
x^1	x^1	x^4	{x^2, x^3}	0
x^2	x^4	x^1	x^2	1
x^3	x^4	-	x^3	1
x^4	x^1	x^4	-	2

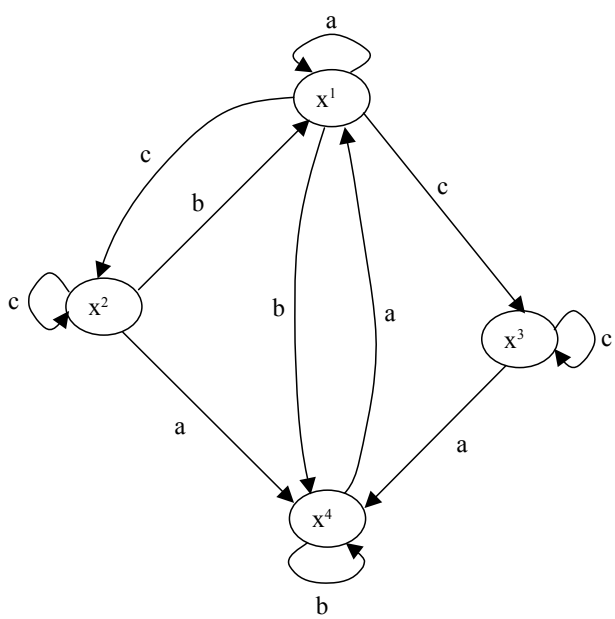

Figure 2-1. State flow diagram of the machine Σ

Table 2-2. Transition table of the machine $\Sigma_{|s}$

	a	b	c	Y
x^1	x^1	x^4	{x^2, x^3}	0
x^2	x^1	x^4	x^2	1
x^3	x^1	-	x^3	1
x^4	x^1	x^4	-	2

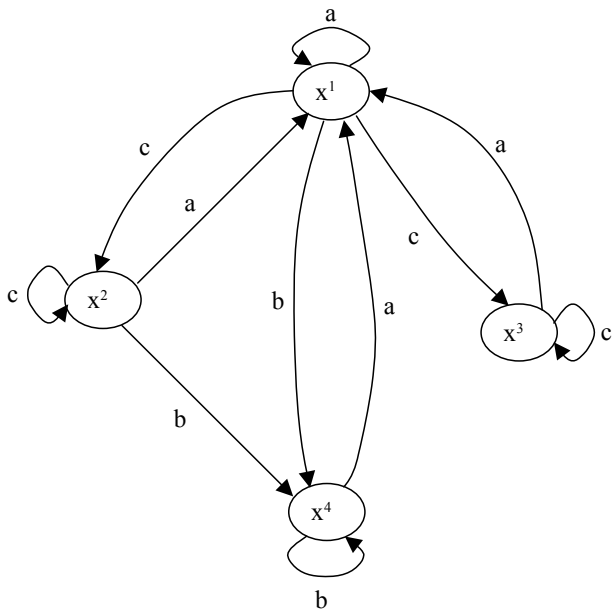

Figure 2-2. State flow diagram of the machine $\Sigma_{|s}$

Let $X^r = \{x^1, x^2, x^3\}$, then $U = \{a, c\}$. Given $u = a$, the next stable states set is $s[X^r, a] = \{x^1\}$. The burst set is $B(X^r, u) = \{\beta_1, \beta_2\} = \{0, 020\}$ and the two burst equivalent subsets of $s[X^r, a]$ are: $S_1 = \{x^1\}$ and $S_2 = \{x^1\}$. The state in S_1 can be reached via burst $\beta_1=0$ and the state in S_2 can be reached via burst $\beta_2=020$, so S_1 and S_2 are two burst equivalent sets.

From the above example, we notice that those states in a burst equivalent set of an asynchronous machine cannot be distinguished from each other by an external observer. So, we need a new method to deal with this kind of situation.

Definition 2-9 Let $X^r := \{x^{i(1)}, ..., x^{i(m)}\}$ be a set of states of the machine Σ, assume they have a non-empty set U of common input characters, and let $u \in U$ be a character. The asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ is detectable at the set pair (X^r, u) if it is possible to determine from input/output data whether all outcomes $s[X^r, u]$ have reached their next stable state; if so, the set of transitions from (X^r, u) to $s[X^r, u]$ is called a stable and detectable transition set. ♦

It was shown in Geng and Hammer (2005) that a stable transition is detectable if and only if its burst switches output characters in its last step. When dealing with detectability of sets of states, the situation is somewhat more complicated. The outcomes of a state-set-input pair (X^r, u) form a set of states $s[X^r, u]$ and all bursts from the set X^r to $s[X^r,u]$ form a set of bursts $B(X^r, u)$. Even if every burst in $B(X^r, u)$ is detectable individually, it is still possible that one cannot determine whether the machine has reached its next stable state. For instance, consider two bursts in $B(X^r, u)$: $\beta(x_1, X^r, u)$ and $\beta(x_2, X^r, u)$. The burst $\beta(x_1, X^r, u)$ is a strict prefix of the burst $\beta(x_2, X^r, u)$, say $\beta(x_1, X^r, u) = y_1y_2y_3$ and $\beta(x_2, X^r, u) = y_1y_2y_3y_4y_5y_3$. Clearly, then, at the point where the burst $\beta(x_1, X^r, u)$ ends, it is not possible to tell whether the machine has reached its next stable combination, as the machine might actually be on its way to the state x_2. This discussion leads us to the following statement.

Proposition 2-10 Let $X^r := \{x^{i(1)}, ..., x^{i(m)}\}$ be a set of states of the machine Σ, and assume they have a non-empty set U of common input characters. For an element $u \in U$, let $s[X^r, u]$ be the set of all possible next stable states, where s is the stable transition function of the machine Σ. Let $B(X^r, u)$ be the set of all possible bursts generated by the pair (X^r, u). The asynchronous machine Σ is detectable at the set pair (X^r, u) if and only if the following conditions are satisfied:

(a) $\beta_{-1} \neq \beta$ for all $\beta \in B(X^r, u)$;

(b) β is not a strict prefix of β' for any $\beta, \beta' \in \beta(X^r, u)$.

Proof. The first condition has been proved in Geng and Hammer (2005). Let us examine the second condition. The first condition guarantees the detectability of the end of each burst in $B(X^r, u)$, so the only confusion is from other bursts in the set $B(X^r, u)$. Consider two bursts β_i, β_j

$\in B(X^r, u)$, where β_i leads to the state x^i, while β_j leads to the state x^j. Assume first that Σ is detectable at the pair (X^r, u). By contradiction, assume that β_i is a strict prefix of β_j, i.e.,

$$\beta_i = y_1 y_2 \ldots y_{k-1} y_k$$

and

$$\beta_j = y_1 y_2 \ldots y_{k-1} y_k y_{k+1} \ldots y_\ell \text{ , where, } y_k \neq y_{k-1} \text{ and } y_\ell \neq y_{\ell-1}.$$

Once the change from y_{k-1} to y_k is observed, it is not possible to determine whether the machine Σ has reached the next stable state x^i or whether it is still in the transition to next stable state x^j. Thus the machine Σ cannot be detectable at (X^r, u), a contradiction. This shows that condition (b) must be valid whenever Σ is detectible at (X^r, u).

Conversely, assume that conditions (a) and (b) are both valid, and let $\beta \in B(X^r, u)$ be a burst, and let $X' \subset s[X^r, u]$ be the set of all states to which the burst β leads. Now, since β is not a strict prefix of any other burst in $B(X^r, u)$, it follows that, at the end of the burst β, the machine Σ must be at one of the states of the set X', say the state x' (i.e., Σ cannot be on its way to other states). Furthermore, since $\beta_{-1} \neq \beta$, the end of the burst β can be determined, and whence it can be determined that Σ has reached a stable combination with a state of X' (note that it cannot be determined from the burst which state of X' has been reached). This completes our proof. ♦

For example, consider the machine Σ with transition table of Table 2-1, let $X^r = \{x^1, x^2\}$, then $U = \{a, b, c\}$. Let us check the detectability of the combination (X^r, a). The next stable states set $s[X^r, a] = \{x^4\}$. The burst from the state x^1 to the state x^4 is $\beta_1 = 02$ and the burst from the state x^2 to the state x^4 is $\beta_2 = 102$. The burst set $\beta(x^4, X^r, a) = \{\beta_1, \beta_2\}$. Since the burst β_1 is not a strict prefix of β_2, and vice versa, the transition from (X^r, a) to x^4 is detectable.

2.2 Generalized Machines, States and Functions

The next notion is central to our discussion. It is sometimes convenient to consider certain sets of states of a machine as one quantity. This is convenient, for example, in cases where the available data at a certain point in time does not permit us to distinguish between these states. This leads us to the following notion of a generalized machine.

Definition 2-11 Let Σ be a machine with the state set X and input set A, let $S(\beta)$ be a burst equivalent set with respect to u of the machine Σ containing more than one state, and let Ξ be a set disjoint from X, and let $\Phi : P(X) \to \Xi$ be a function. Associate with $S(\beta)$ the element $x_b := \Phi(S(\beta))$; we call x_b a burst state. The set Ξ is then called the set of potential burst states and Φ is called the burst state assignment function. Let A be the set of all common input characters of the states in $S(\beta)$. Then, the set of all valid pairs of x_b is given by $\{(x_b, a) : a \in A\}$. The set A is also called the valid input set of the burst state x_b. Let $X_b \subset \Xi$ be the set of all burst states of the machine Σ. The generalized state set $\tilde{X}$ of Σ is the union $X \cup X_b$. The burst equivalent set $S(\beta)$ represented by a burst state x_b is also recorded as $S(x_b)$. ♦

Let $\Sigma_{|s} = (A, Y, X, x_0, s, h)$ be the stable state machine of an asynchronous machine Σ. Let $X^r := \{x^{i(1)}, ..., x^{i(m)}\} \subset X$ be a set of states of the machine Σ, and assume they have a non-empty set U of common input characters. For an element $u \in U$, let $s[X^r, u]$ be the set of all possible next stable states. Let $B(X^r, u) = \{\beta_1, \beta_2, \ldots, \beta_\ell\}$ be the set of all possible bursts generated by the transition $(X^r, u) \to s[X^r, u]$, and let $S(\beta_1), S(\beta_2), \ldots, S(\beta_\ell)$ be the burst equivalent sets in $s[X^r, u]$. For each $i = 1, 2, ..., \ell$, we distinguish between two cases:

1) The set $S(\beta_i)$ contains a single state $x \in X$. Then, we identify $S(\beta_i)$ with the state x.

2) The set $S(\beta_i)$ contains more than one state. Then, we associate with $S(\beta_i)$ a burst state, which represents the fact that these states are indistinguishable in this transition.

For the second case, let A_i be the set of all common input characters of $S(\beta_i)$, $i = 1, 2, \ldots, \ell$. Note that A_i cannot be empty, since at least A_i contains the element u.

Definition 2-12 Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with a generalized state set $\tilde{X} = X_b \cup X$, where X is the regular state set of Σ and X_b is the burst state set of the machine Σ. We build now a generalized stable transition function $s_g : \tilde{X} \times A \rightarrow P(\tilde{X})$ as follows:

1) For all states $x \in X$ and all input characters $u \in A$, set $s_g(x, u) := s(x, u)$.

2) For a burst state $x \in X_b$, let $U(x) \subset A$ be the set of all input characters that form valid pairs with x. Let S(x) be the burst equivalent set represented by the burst state x. For an input character $a \in U(x)$, let $\Delta^1, \Delta^2, \ldots, \Delta^m$ be the set of all burst equivalent subsets of $s[S(x), a]$. If Δ^i contains more than one state, then let x^i be the burst state associated with the set Δ^i; otherwise Δ^i is represented by its only state x^i, $i = 1, \ldots, m$. Then, $s_g(x, a) := \{x^1, \ldots, x^m\}$. ♦

Note that the next stable states of a machine can be a combination of burst states and regular states.

In the next discussion, I will give a more specific algorithm to build the generalized stable transition function. As a practical process, the algorithm should avoid getting involved into infinite loops. Thus, before the construction we need to make sure about two issues: a) the process of building the generalized stable transition function includes finite steps; b) there is no infinite cycles created in the construction. Since every burst equivalent set $S(\beta)$ is a subset of the state set X, for an asynchronous machine Σ with n regular states, the maximum number of

subsets in X is 2^n. Hence, the number of burst equivalent sets is equal or less than 2^n. Namely, the number of burst states generated in the machine Σ is finite. Then the first requirement is guaranteed. In the previous discussion, we have excluded asynchronous machine with infinite cycles. So, any transition starting from a regular state of a machine in this paper ends at the next stable states. Similarly, under the definitions of the generalized state and generalized stable transition function, for each valid state-input pair there is one or more next stable states. Hence, each transition starting from a generalized state ends at the next stable states, i.e., no infinite cycles will be created in the process of defining the generalized stable transition function s_g.

Consider an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ with stable state machine $\Sigma_{|s} = (A, Y, X, x_0, s, h)$. Let $\tilde{X} = X_b \cup X$ be the generalized state set and let $X_b := \{\xi_1, \xi_2, \ldots, \xi_t\}$ be the burst state set of Σ. For every burst state $\xi_c \in X_b$, let $A_b := \{a_1, a_2, \ldots, a_{g(c)}\}$ be the valid input set of ξ_c. For every valid pair (ξ, a), $\xi \in X_b$ and $a \in A_b$, let $S(\xi)$ be the set of regular states represented by ξ and let $s[S(\xi), a)$ be the set of all possible next stable states of $[S(\xi), a]$. Assume that $\Sigma_{|s}$ have ρ critical races $(r_1, v_1), (r_2, v_2), \ldots, (r_\rho, v_\rho)$, and let $T(r_i, v_i) := \{r_i^1, r_i^2, \ldots, r_i^{m(i)}\} \subset X$ be the set of all outcomes of the critical race (r_i, v_i), $i = 1, \ldots, \rho$. We build the generalized stable transition function s_g with the following algorithm.

Algorithm 2-13 Consider an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ with stable state machine $\Sigma_{|s} = (A, Y, X, x_0, s, h)$. Let $\tilde{X} = X_b \cup X$ be the generalized state set, where X_b is the burst state set and X is the regular state set. Assume that $\Sigma_{|s}$ has ρ critical races (r_1, v_1), $(r_2, v_2), \ldots, (r_\rho, v_\rho)$, and let $T(r_i, v_i) := \{r_i^1, r_i^2, \text{ of the } \ldots, r_i^{m(i)}\} \subset X$ be the set of all outcomes

critical race (r_i, v_i), $i = 1, \ldots, \rho$.

For every state $x \in X$ and $u \in A$, if $s(x, u)$ is a single state, then set $s_g(x, u) := s(x, u)$. Set $X_b := \varnothing$ and let $K := \{(r_1, v_1), (r_2, v_2), \ldots, (r_\rho, v_\rho)\}$ be the set of all critical race pairs of the machine Σ. Set $i := 1$ and run the following steps:

Step 1:

a) Consider the i-th element (r_i, v_i) of the set K. If $i \leq \rho$, then let $X_i := \{r_i^1, r_i^2, \ldots, r_i^{m(i)}\}$ be the outcomes of the critical race (r_i, v_i).

b) If $i > \rho$, then the i-th element (r_i, v_i) of the set K is a burst-state-input pair created in Step 3 of a previous cycle of the algorithm. Let $S(r_i)$ be the state set associated with the burst state r_i. Let $X_i := s[S(r_i), v_i]$ be the set of all possible next stable states of the set of states $S(r_i)$ with the input character v_i.

Step 2:

Set $j = 0$. Partition the set X_i into its burst equivalent subsets $T_1, T_2, \ldots, T_t$ with respect to the input character v_i, and denote by $T := \{T_1, T_2, \ldots, T_t\}$ the corresponding class of subsets. Let Z be the set consisting of all subsets T_j that contain a single state; if there are no such subsets in T, then set $Z := \varnothing$. Denote by $S^i := T \setminus Z$ the corresponding difference set. If $S^i = \varnothing$, then set $k := 0$ and go to Step 4. Otherwise, Let $S_1^i, S_2^i, \ldots, S_k^i$ be the members of S^i.

Step 3 :

Set $j := j+1$ and check the set S_j^i as follows. Let Ξ be the set of potential burst states and let $\Phi : P(X) \to \Xi$ be the burst state assignment function.

If $\Phi(S_j^i) \notin X_b$, then proceed as follows; otherwise, go to b).

Add the burst state $x_j^i := \Phi(S_j^i)$ to X_b, i.e., set $X_b := X_b \cup x_j^i$.

Let $A_c := \{u_1, u_2, \ldots, u_{g(i,j)}\}$ be the valid input set of the burst state x_j^i. Let $\eta := \#K$ be the number of elements of the set K. Add to K the elements $(r_{\eta+\alpha}, v_{\eta+\alpha}) := (x_j^i, u_\alpha)$, $\alpha = 1, \ldots, g(i,j)$.

Add the burst state $\Phi(S_j^i)$ to Z.

If $j < k$, then go back to Step 3.

Step 4.

Set $s_g(x_i, u_i) := Z$.

Step 5.

If $i < \#K$, then set $i = i + 1$ and go back to step 1. Otherwise, terminate the algorithm. ♦

The set $\tilde{X} := X \cup X_b$ is the generalized state set of the machine Σ. The generalized stable transition function is $s_g : \tilde{X} \times A \to P(\tilde{X})$.

According to definition of the *burst of a string*, the last character of a burst is the output value of the system for the corresponding state. Consequently, all states in a burst equivalent set have the same output value. This implies that the following is true.

Lemma 2-14 The output value of a burst state x is the output value of any state in the corresponding burst equivalent set $S(x)$. ♦

Definition 2-15 Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with a generalized state set $\tilde{X} = X_b \cup X$, where X is the regular state set of Σ and X_b is the burst state set of the machine Σ. Let x be a generalized state of the machine Σ. The generalized output function $h_g : \tilde{X} \times A \to Y$ of Σ is defined as follows:

1) For all states $x \in X$, set $h_g(x) := h(x)$;

2) For all burst state $x \in X_b$, let $S(x)$ be the burst equivalent set that is associated with the burst state x. Set

$hg(x) := h(x'), x' \in S(x)$. ♦

For example, consider the machine Σ with transition table of Table 2-1, which has one critical race $s(x^1, c) = \{x^2, x^3\}$. Using Algorithm 2-13, we can get the burst state set $X_b = \{x^5\}$ and x^5 represents the subset $\{x^2, x^3\}$. The generalized stable recursion function s_g and the generalized output function h_g of the machine Σ can also be defined (Table 2-3).

Table 2-3. Stable transition table of the generalized Σ

	a	b	c	Y
x^1	x^1	x^4	x^5	0
x^2	x^1	x^4	x^2	1
x^3	x^1	-	x^3	1
x^4	x^1	x^4	-	2
x^5	x^1	-	x^5	1

Definition 2-16 Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable state machine $\Sigma_{|s} = (A, Y, X, x_0, s, h)$. Then, $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ is the generalized machine associated with Σ, where $\tilde{X}$ is the generalized state set, s_g is the generalized stable recursion function, and h_g is the generalized output function of the machine Σ. ♦

When an asynchronous machine is enhanced into a generalized machine, it still keeps some properties. We address two properties of the generalized machine in these two statements.

Lemma 2-17 Given an asynchronous machine Σ with a state set X, which contains finite number of states. Then the associated generalized machine Σ_g also has a generalized state set $\tilde{X}$ with finite number of states.

Proof. Let us suppose the machine Σ has a state set $X = \{x^1, x^2, \ldots, x^n\}$ and the generalized machine Σ_g has a generalized state set $\tilde{X} = \{x^1, x^2, \ldots, x^{n+t}\}$. From the definition of burst equivalent set, any burst equivalent set $S(\beta)$ is a subset of the state set X. Then, the maximum number of burst equivalent sets cannot be larger than the number of subsets of X, i.e., 2^n, and hence is finite. ♦

Lemma 2-18 If the machine Σ has no infinite cycles, neither does the machine Σ_g.

Proof. Assume the machine Σ has no infinite cycles but the generalized machine Σ_g, which is derived from the machine Σ, has one infinite cycle χ of length i, where $i > 1$. Suppose that i generalized states $x_1, x_2, ..., x_i$ are involved in this infinite cycle χ. The states $x_1, x_2, ..., x_i$ may be regular states or burst states. Let us consider the following two cases: i) If all these i states are regular states. Then it implies that the machine Σ has at least one infinite cycle χ'. And the infinite cycle involves the i states $x_1, x_2, ..., x_i$ of the machine Σ. It conflicts with the assumption that the machine Σ has no infinite cycles. ii) If in the i states $x_1, x_2, ..., x_i$ there is at least one burst state x_p, where $1 \le p \le i$. Suppose that the underling regular states of the burst state x_p are $x_p^1, ..., x_p^k$. Then the infinite cycle χ actually involves the following regular states: $x_1, ..., x_{p-1}, x_p^j, x_{p+1}, ..., x_i$, where $1 \le j \le k$. Hence there is an infinite cycle χ'' that involves i regular states of the machine Σ. It conflicts with the assumption that the machine Σ has no infinite cycles. This completes the proof. ♦

2.3 Observer

As depicted in Figure 1-1, we build an output feedback loop with a controller C, which is also an asynchronous machine. Specifically, this controller C is composed of two asynchronous machine: an observer B and a state-feedback control unit F (Figure 2-3).

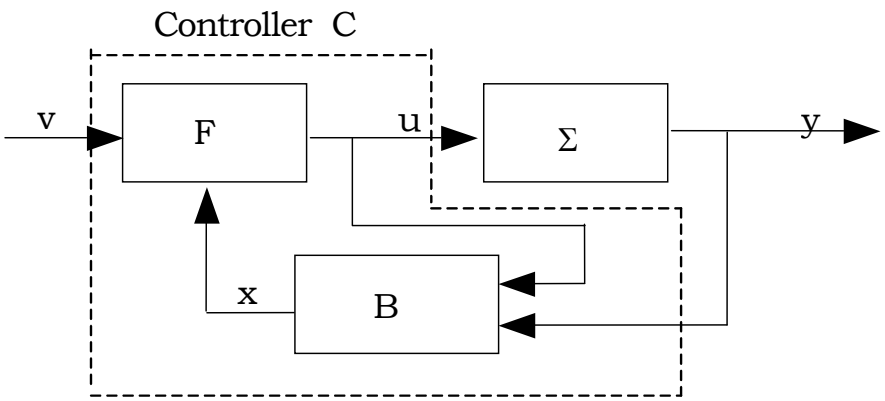

Figure 2-3. Control configuration for the closed-loop system Σ_c

Here, the observer B estimates the uncertainty caused by critical races with the input/output information of Σ and generates estimate state of Σ to feed F. With the external input of the whole system and the estimate state of Σ, the control unit F generates a sequence of input to drive Σ to match the model. We denote the controller C with (F, B).

We use the observer in a way that is similar as it is used in other branches of control theory. Specifically, the observer here is an asynchronous input/state machine, which has two functions: a) check if the asynchronous machine Σ has reached its next stable state; b) use the input/output information to estimate the current state of Σ.

Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be the generalized machine derived from Σ. Similarly as in Geng and Hammer (2005), we can build an observer that reproduces all stable and detectable transitions of the machine Σ_g. The observer for Σ_g is an input/state machine $B = (A \times Y^*, \tilde{X}, Z, z_0, \sigma, I)$ with two inputs: the input character $u \in A$ of Σ_g and the output burst $\beta \in Y^*$ of Σ_g; the state set Z is identical to the generalized state set $\tilde{X}$, and the initial condition is identical to that of Σ_g, i.e., $z_0 = x_0$. The recursion function $\sigma : Z \times A \times Y^* \to Z$ of B is

constructed as follows. First, using the generalized stable recursion function s_g, define the function $\lambda : Z \times A \times \{0, 1\} \to Z$ by setting

$$\lambda(z, u, a) := \begin{cases} s_g(z, u) & \text{if } a = 1; \\ z & \text{if } a = 0. \end{cases} \tag{2-2}$$

Now, assume that the machine Σ_g is in a stable combination (x, u_{i-1}), when the input character changes to u_i, where (x, u_i) is also a detectable pair. The change of the input character may give rise to a chain of transitions of Σ_g. Let $k \geq i$ be a step during this chain of transitions, let β_k be the burst of Σ_g from step i to step k, and let u_k be the input character of Σ_g at step k. Since fundamental mode operation requires that the input character be kept constant during a chain of transitions, we have $u_k = u_i$. Define

$$\sigma(x, u_k, \beta_k) := \begin{cases} s(x, u_k) & \text{if } \beta_k = \beta(x, u_k); \\ x & \text{otherwise.} \end{cases} \tag{2-3}$$

Let z_k be the state of the observer B at the step k, while ω_k be the output of B. The observer B is then an input/state machine defined by the recursion

$$B := \begin{cases} z_{k+1} = \sigma(z_k, u_k, \beta_k) \\ \omega_k = z_k \end{cases} \tag{2-4}$$

The observer B is a stable state machine.

To describe the operation of the observer, assume that the observer switched to the generalized state x immediately after Σ_g has reached the stable combination (x, u_{i-1}). Let $p \geq i$ be the step at which the chain of transitions from (x, u_i) to the next stable state $x' = s_g(x, u_i)$ terminates; then, $\beta_p = \beta(x, u_i)$. As the pair (x, u_i) is detectable, it follows by the definition of σ that the output of the observer B switches to the state x' at the step $p+1$.

We can now summarize the implications of our recent observations on the control configuration Figure 1-1. Fundamental mode operation requires the output of the controller C to remain constant while the system Σ_g is in transition. In order to fulfill this requirement, it must be possible for the controller C to detect the point at which Σ_g has completed its transition process. As discussed above, the output of the observer B switches to the state that represents the next generalized stable state of Σ_g immediately after Σ_g has reached that state; this signifies the end of the transition process and indicates the most recent stable state of Σ_g. In this way, the observer B helps create an environment in which the machine Σ_g can be controlled in fundamental mode operation.

CHAPTER 3
REACHABILITY OF A GENERALIZED MACHINE

The occurrence of critical races in an input/output asynchronous machine causes the lacks of information about the exact state of the machine. We use the concept of generalized states to deal with this uncertainty and keep a machine operate in fundamental mode. In this chapter, we use generalized states to characterize the reachability properties of an asynchronous machine with critical races.

First, let me introduce some important concepts that will be used in latter part of this chapter.

3.1 Generalized Reachability Matrix

Definition 3-1 Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be the generalized machine associated with Σ, where $\tilde{X} = \{ x^1, \ldots, x^n, x^{n+1}, \ldots, x^{n+t}\}$ is the generalized state set. The generalized one-step reachability matrix $R(\Sigma_g)$ is defined as a $(n+t)\times(n+t)$ matrix with entry R_{ij}, where R_{ij} is the set of all characters $a \in A$ for which $x^j \in s_g(x^i, a)$ and for which the transition $x^i \rightarrow x^j$ is a detectable transition. If there is no such character a, then set $R_{ij} := N$, where N is a character not in the alphabet A.♦

Note that when the generalized machine Σ_g is equal to the machine Σ (i.e., when there are no burst states), then the generalized one-step reachability matrix reduces to the one-step reachability matrix $R(\Sigma)$.

In view of the earlier discussion in Geng and Hammer (2005), only transitions that are both stable and detectable can be used when constructing a controller. The stability of the transition is guarantied by the generalized stable recursion function of the controlled machine Σ.

However, the detectability of each transition needs to be checked according to proposition 2-10 in the construction of the generalized one-step reachability matrix. Therefore, each entry of the generalized one-step reachability matrix characterizes if the machine Σ_g can go from one generalized state to another through a stable and detectable transition.

Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be the generalized machine associated with Σ, where $\tilde{X} = \{ x^1, \ldots, x^n, x^{n+1}, \ldots, x^{n+t}\}$ is the generalized state set and $X_b = \{ x^{n+1}, \ldots, x^{n+t} \} \subset \tilde{X}$ is the burst state set. According to Definition 3-1, we can obtain the one-step reachability matrix $R(\Sigma)$ of the machine Σ. For the generalized machine Σ_g, the construction the generalized one-step reachability matrix $R(\Sigma_g)$ contains two tasks: 1) Determine the necessary burst states; and 2) Add to the reachability matrix rows and columns corresponding to the burst states. Then, we can divide $R(\Sigma_g)$ into 4 blocks

$$R(\Sigma g) = \left[\begin{array}{c|c} R_{11} & R_{12} \\ \hline R_{21} & R_{22} \end{array}\right],$$

where, R_{11} is $n \times n$; R_{12} is $n \times t$, R_{21} is $t \times n$, and R_{22} is $t \times t$. The matrix R_{11} describes one-step deterministic transitions among regular states of Σ, while R_{22} describes one-step transitions among burst states. The submatrix R_{12} represents one-step transitions from regular states to burst states, while R_{21} represents one-step transitions from burst states to regular states.

Example 3-2 Consider the machine Σ with transition table of Table 2-1, which has the input alphabet $A = \{a, b, c\}$, the output alphabet $Y = \{0, 1, 2\}$, and the state set $X = \{x^1, x^2, x^3, x^4, x^5\}$. There is a critical race pair (x^1, c) in the machine Σ.

The generalized machine Σ_g derived from the Σ has a generalized state set $\tilde{X} = \{x^1, ..., x^5\}$, and it can be depicted as follows.

Table 3-1. Stable state transition table of the machine Σ_g

	a	b	c	Y
x^1	x^1	x^4	x^5	0
x^2	x^1	x^4	x^2	1
x^3	x^1	-	x^3	1
x^4	x^1	x^4	-	2
x^5	x^1	-	x^5	1

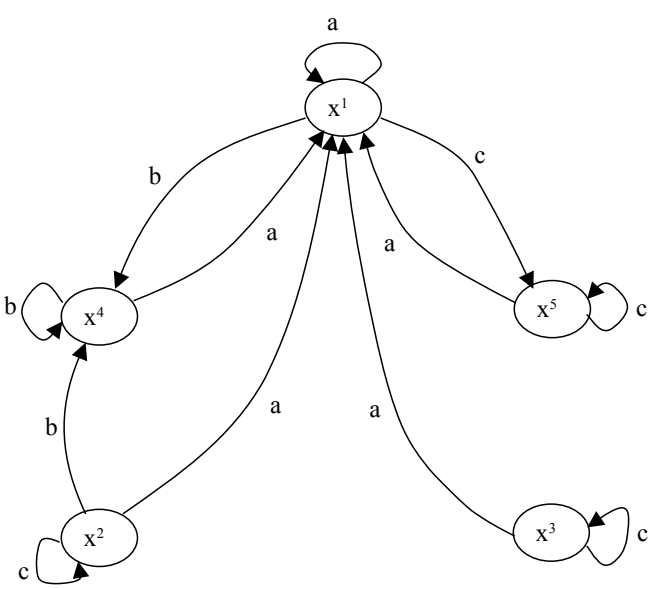

Figure 3-1. State flow diagram of the machine Σ_g

According to Definition 3-1, the one-step reachability matrix of the original machine Σ is

$$R(\Sigma) = \begin{bmatrix} a & c & c & b \\ a & c & N & b \\ a & N & c & N \\ a & N & N & b \end{bmatrix}$$

and the generalized one-step reachability matrix of the machine Σ_g is

$$R(\Sigma_g) = \begin{bmatrix} a & N & N & b & c \\ a & c & N & b & N \\ a & N & c & N & N \\ a & N & N & b & N \\ a & N & N & N & c \end{bmatrix}.$$

The submatrix $R_{22} = [c]$ in the matrix $R(\Sigma_g)$ describes the stable and detectable transitions inside the burst state set X_b. ♦

After obtaining the generalized stable recursion function s_g we can get the generalized one-step reachability matrix $R(\Sigma_g)$ as above. Similar to the one-step transition matrix in Venkatraman and Hammer (2004), we define some operations on the generalized one-step reachability matrix $R(\Sigma_g)$. Based on these operations we can obtain the overall view of reachable states in the generalized machine Σ_g and the information about how to approach the destination of any transition. The latter information is very useful in the construction of the controller of the closed loop system.

Definition 3-3 Let A^* be the set of all strings of characters of the alphabet A and let w_i be a subset of A^* or the character N, $i = 1, 2$. The operation $\cup\!\!\!/$ of unison is defined by

$$w_1 \cup\!\!\!/\, w_2 := \begin{cases} w_1 \cup w_2 & \text{if } w_1, w_2 \in A^* \\ w_1 & \text{if } w_1 \in A^* \text{ and } w_2 = N \\ w_2 & \text{if } w_1 = N \text{ and } w_2 \in A^* \\ N & \text{if } w_1 = w_2 = N. \end{cases} \tag{3-1}$$

The unison $C := A \curlyvee B$ of two $n \times n$ matrices A and B, whose entries are either subsets of A^* or the character N, is defined entrywise by $C_{ij} := A_{ij} \curlyvee B_{ij}$, $i, j = 1, ..., n$. ♦

Note that N takes the role of zero.

For example, given two 3×3 matrices $A = \begin{bmatrix} a & b & N \\ b & N & N \\ a & a & c \end{bmatrix}$ and $B = \begin{bmatrix} N & b & N \\ c & N & N \\ a & b & b \end{bmatrix}$, the unison of A and B is

$$C := A \curlyvee B = \begin{bmatrix} a & b & N \\ \{b,c\} & N & N \\ a & \{a,b\} & \{b,c\} \end{bmatrix}.$$

Definition 3-4 Let A^* be the set of all strings of characters of the alphabet A and let w_1, w_2 be two subsets of A^* or the character N. Concatenation of elements $w_1, w_2 \in A^* \cup N$ is defined by

$$\text{conc}(w_1,w_2) := \begin{cases} w_2 w_1 & \text{if } w_1, w_2 \in A^* \\ N & \text{if } w_1 = N \text{ or } w_2 = N. \end{cases} \tag{3-2}$$

Let $W = \{w_1, w_2, \ldots, w_q\}$ and $V = \{v_1, v_2, \ldots, v_r\}$ be two subsets, whose elements are either subsets of A^* or the character N. Define

$$\text{conc}(W, V) := \curlyvee_{\substack{i = 1, \ldots, q \\ j = 1, \ldots, r}} \text{conc}(w_i,v_j) \,. \; ♦ \tag{3-3}$$

For instance, consider two subsets $W = \{a,N,\{b,c\}\}$ and $V = \{N,a,\{a,b\}\}$. The concatenation of W and V is

$$\text{conc}(W, V) = \{a,aa,\{aa,ba\},N,\{a,b\},\{b,c\},\{ab,ac\},\{ab,bb,ac,bc\}\}.$$

Definition 3-5 Let C and D be two $n \times n$ matrices whose entries are either subsets of A^* or the character N. Let C_{ij} and D_{ij} be the (i,j) entries of the corresponding matrices. Then, the product $Z := CD$ is an $n \times n$ matrix, whose (i,j) entries Z_{ij} is given by

$$Z_{ij} := \bigcup_{k=1}^{n} \mathrm{conc}(C_{ik}, D_{kj}),\ i,j = 1, ..., n. \ \blacklozenge \tag{3-4}$$

For example, consider two 3×3 matrices $A = \begin{bmatrix} a & b & N \\ b & N & N \\ a & a & c \end{bmatrix}$ and $B = \begin{bmatrix} N & b & N \\ c & N & N \\ a & b & b \end{bmatrix}$. Then the product of A and B is

$$Z = AB = \begin{bmatrix} cb & ba & N \\ N & bb & N \\ \{ac,ca\} & \{ba,bc\} & bc \end{bmatrix}.$$

Using the operation of product, we can define powers of the generalized one-step reachability matrix by setting

$$R^q(\Sigma_g) := R^{q-1}(\Sigma_g)R(\Sigma_g), \quad q = 2, 3, ... \tag{3-5}$$

Proposition 3-6 All transitions of the matrix $R^q(\Sigma_g)$ are stable and detectable transitions.

Proof. According to the definition of the generalized one-step reachability matrix, each non-zero entry of the matrix refers to a stable and detectable transition. After the operation of product, every non-zero entry of a generalized multi-step reachability matrix refers to a combination of multi-step stable and detectable transitions. This operation does not change either the stability or the detectability of the transitions. Thus, for every entry of the matrix $R^q(\Sigma_g)$, if it is not N, it stands for a stable and detectable transition. ♦

Based on Proposition 3-6, for an integer $q \geq 1$, the matrix $R^q(\Sigma_g)$ describe if the machine can reach one state from another state through exact q stable and detectable transitions. If the (i,j) entry of the matrix $R^q(\Sigma_g)$ is not N, then it is the set of all input strings that can takes the machine Σ_g from the state x^i to the state x^j via a q-step stable and detectable transition. If the (i,j) entry is N, then it is impossible to reach the state x^j from the state x^i in exact q stable and detectable transitions. Though, it might be possible to reach from x^i to x^j in θ stable and

detectable transitions, where $\theta < q$ or $\theta > q$. Since we are also interested in if the machine can reach one state from another in fewer transitions, it is needed to construct a multi-step reachability matrix.

Definition 3-7 Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalized asynchronous machine and let $R(\Sigma_g)$ be the generalized one-step reachability matrix of the machine Σ_g. The generalized q-step reachability matrix is defined by

$$R^{(q)}(\Sigma_g) := \bigvee_{r = 1, \ldots, q} R^r(\Sigma_g), \quad q = 2, 3, \ldots \quad \blacklozenge \tag{3-6}$$

Note that the (i,j) entry of $R^{(q)}(\Sigma_g)$ contains the reachability information from the state x^i to the state x^j. If the (i,j) entry is not N, it consists all strings that may take the machine Σ_g from $x^i \rightarrow x^j$ through stable and detectable transitions in q or fewer steps. It leads to the following statement and its proof is similar to Murphy, Geng and Hammer (2003).

Lemma 3-8 Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalized asynchronous machine with n states and t burst states, and let $R(\Sigma_g)$ be the generalized reachability matrix of the machine Σ_g. Then the following two statements are equivalent:

(i) The generalized state x^j is stably reachable through a detectable transition from the generalized state x^i.

(ii) The (i,j) entry of $R^{(n+t-1)}(\Sigma_g)$ is not N. ♦

Proof. Let A^* be the set of all strings of characters of the alphabet A.

If the first statement is true, namely, the state x^j is stably reachable from x^i, then there is an input string $u' := u_{k-1} \ldots u_1 u_0$, which satisfies $x^j = s_g(x^i, u)$ and the transition from x^i to x^j is

detectable . Here, $u' \in A^*$ and $k := |u'|$. If $|u'| \leq (n + t - 1)$, then take $u := u'$. Thus, the (i,j) entry of $R^{(n+t-1)}(\Sigma_g)$ is $u_{k-1}\ldots u_1 u_0$.

If $|u'| \geq (n + t - 1)$, then we need to show that a shorter string u^* in the string u' still satisfies $x^j = s_g(x^i, u^*)$ and $|u^*| \leq (n + t - 1)$. Define recursively a string of states $x_0, x_1, ..., x_k$, by setting $x_0 := x^i$ and $x_{m+1} := s_g(x_m, u_m)$, for $m = 0, 1, ..., k-1$. This implies $x_k = x^j$. The length of the string $x_0, ..., x_k$, $k+1$ is greater or equal to $(n + t)$. However, there are totally only $(n + t)$ distinct generalized states of the machine Σ_g. So, at least one generalized state must be repeated in the string $x_0, ..., x_k$. Suppose $x_p = x_q$, for $0 \leq p \leq q \leq k$. Remove from u' the string v, which satisfies $x^q = s_g(x^p, v)$. Afterwards the shortened input string

$u'' := u_0 u_1 \ldots u_{p-1} u_q \ldots u_{k-1}\, u_{k-1} \ldots u_q u_{p-1} \ldots u_1 u_0$ (or $u'' := u_{k-1}\ldots u_q$ when $p = 0$).

This shortened u'' still satisfies $x^j = s_g(x^i, u'')$. Keep shortening the input string until an input string u^* of length $|u^*| \leq (n + t - 1)$ still satisfying $x^j = s_g(x^i, u^*)$.

Conversely, if the second statement is true, then it implies that there is an input string $u \in A^*$ of length $|u| \leq (n + t - 1)$. Suppose the (i,j) entry of $R^{(n+t-1)}(\Sigma_g)$ is the string $u := u_{k-1}\ldots u_1 u_0$. There are k input characters in the string u and those characters satisfy the following equation:

setting $x_0 := x^i$ and $x_{m+1} := s_g(x_m, u_m)$, for $m = 0, 1, ..., k-1$. Then we have $x_k = x^j$. According to the definitions of the generalized stable recursion function, the machine Σ_g is only involved into stable transitions here. Meanwhile, in the construction of the generalized reachability matrix $R(\Sigma_g)$, all undetectable transitions are eliminated. Thus, the generalized (n + t

– 1) step reachability matrix $R^{(n+t-1)}(\Sigma_g)$ only contains detectable transitions. So, the generalized state x^j is stably reachable from the generalized state x^i. ♦

Therefore, all possible stable and detectable transitions for the machine Σ_g can be found in the matrix $R^{(n+t-1)}(\Sigma_g)$, i.e., the generalized $(n + t - 1)$ step reachability matrix characterizes the reachability property of the machine Σ_g with n states and t burst states.

Definition 3-9 Let $R(\Sigma_g)$ be the generalized one-step reachability matrix of the machine Σ_g, which has $n + t$ generalized states. The generalized stable reachability matrix of the machine Σ_g is $\Gamma(\Sigma_g) := R^{(n+t-1)}(\Sigma_g)$. ♦

Example 3-10 Consider the machine Σ and the generalized machine Σ_g of Example 3-2. Σ_g has a generalized state set $\tilde{X} = \{x^1, x^2, x^3, x^4, x^5\}$ and $(n + t - 1) = 4$.

Raise the power of the $R(\Sigma_g)$ as follows:

$$R^2(\Sigma_g) = \begin{bmatrix} \{aa,ba,ab,bb,ac\} & N & N & ac & \{ca,cb,cc\} \\ ba & \{cc,ca\} & N & \{ac,aa\} & N \\ \{aa,ba,ac\} & N & cc & N & ca \\ \{ab,bb,ba\} & \{cc,ca\} & N & \{ac,aa\} & cb \\ \{aa,ba,ac\} & ca & N & \{aa,ac\} & \{ca,cc\} \end{bmatrix}.$$

After a stable transition, repeat applying the same input character will not change the state of the machine. Thus, all same consecutive input character can be replaced by one character. For instance, the input string “aa” can be replaced by “a” and it will not affect the stable transitions of the machine. Hence, we obtain

$$R^2(\Sigma_g) = \begin{bmatrix} \{a,b,ba,ab,ac\} & N & N & ac & \{c, ca,cb\} \\ ba & \{c,ca\} & N & \{a,ac\} & N \\ \{a,ba,ac\} & N & c & N & ca \\ \{b,ab,ba\} & \{c,ca\} & N & \{a,ac\} & cb \\ \{a,ba,ac\} & ca & N & \{a,ac\} & \{c,ca\} \end{bmatrix}.$$

Continue raising the power of the $R(\Sigma_g)$ until the $(n + t - 1) = 4$. Then we have

$$R^4(\Sigma_g) = \begin{bmatrix} \left\{\begin{matrix} a,ab,ac,aba,aca \\ abab,abac,acab,acac \end{matrix}\right\} & N & N & \left\{\begin{matrix} b,ba,bab,bac \\ baba,baca \end{matrix}\right\} & \left\{\begin{matrix} c,ca,cab,cac \\ caba,caca \end{matrix}\right\} \\ \left\{\begin{matrix} a,ab,ac,aba,abc \\ aca,abab,abac \end{matrix}\right\} & c & N & \left\{\begin{matrix} b,ba,bc,bab,bac \\ baba,babc,baca \end{matrix}\right\} & \left\{\begin{matrix} ca,cab,cac,caba \\ cabc,caca \end{matrix}\right\} \\ \left\{\begin{matrix} a,ac,aba,aca \\ abac,acac \end{matrix}\right\} & N & c & \{ba,bc,bac,baca,baba\} & \{ca,cac,caba,caca\} \\ \left\{\begin{matrix} a,ab,aba,aca \\ abab,acab \end{matrix}\right\} & N & N & \{b,ba,bab,baba,baca\} & \{ca,cab,caba,caca\} \\ \left\{\begin{matrix} a,ac,aba,aca \\ abac,acac \end{matrix}\right\} & N & N & \{ba,bac,baba,baca\} & \{ c,ca,cac,caba,caca\} \end{bmatrix}$$

According to Definition 3-7 and Definition 3-9, we obtain the generalized stable reachability matrix $\Gamma(\Sigma_g)$ of the machine Σ_g as follow:

$$\Gamma(\Sigma_g) = \begin{bmatrix} \begin{Bmatrix} a,ab,ac,aba,aca \\ abab,abac,acab,acac \end{Bmatrix} & N & N & \begin{Bmatrix} b,ba,bab,bac \\ baba,baca \end{Bmatrix} & \begin{Bmatrix} c,ca,cab,cac \\ caba,caca \end{Bmatrix} \\ \begin{Bmatrix} a,ab,ac,aba,abc \\ aca,abab,abac \end{Bmatrix} & c & N & \begin{Bmatrix} b,ba,bc,bab,bac \\ baba,babc,baca \end{Bmatrix} & \begin{Bmatrix} ca,cab,cac,caba \\ cabc,caca \end{Bmatrix} \\ \begin{Bmatrix} a,ac,aba,aca \\ abac,acac \end{Bmatrix} & N & c & \{ba,bc,bac,baca,baba\} & \{ca,cac,caba,caca\} \\ \begin{Bmatrix} a,ab,aba,aca \\ abab,acab \end{Bmatrix} & N & N & \{b,ba,bab,baba,baca\} & \{ca,cab,caba,caca\} \\ \begin{Bmatrix} a,ac,aba,aca \\ abac,acac \end{Bmatrix} & N & N & \{ba,bac,baba,baca\} & \{c,ca,cac,caba,caca\} \end{bmatrix}. \blacklozenge$$

3.2 Common-output Generalized States

As we have mentioned before, if the outcomes of a critical race (r, v) can be divided into more than one burst equivalent set, then $s_g(r, v)$ consists of more than one generalized state. This situation is shown in the generalized one-step reachability matrix as one input string appears more than once in different entries of a single row. Consider both this fact and the Lemma 3-8, we have the statement below.

Proposition 3-11 Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalized asynchronous machine with n states and t burst states, and let $\Gamma(\Sigma_g)$ be the generalized stable reachability matrix of the machine Σ_g. Then the following two statements are equivalent for all input strings $u \in A^+$ and for all $j = 1, ..., n + t$.

(i) Applying u at the generalized state x^i generates a critical race.

(ii) The string u appears in more than one entry of row i of the matrix $\Gamma(\Sigma_g)$. ♦

Note that the above conclusion is similar with the Proposition 4-16 in Venkatraman and Hammer (2004).

After using burst states to represent subsets of states which have the same output value and same burst, we can see there are still critical races in the machine on the generalized state base. Some input strings may be repeated in more than one entries of a row of the generalized stable reachability matrix (Example 3-10). That is caused by the existence of critical races.

In the present discussion, the machine $\Gamma(\Sigma_g)$ is an input/output machine. Thus, what matters to the user are the output value but not the state of the machine itself. Next we check if the machine can be led from different outcomes of the critical races to the same output value. In Venkatraman and Hammer (2004), if the machine can be led from different outcomes of the critical races to the same state, then it means the existence of a feedback trajectory. Here, we can loose the restriction to a subset of states which have the same output value. They can be also called "common-output generalized states".

Definition 3-12 Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalized asynchronous machine with a generalized state set $\tilde{X} = X_b \cup X$. Assume that the machine Σ_g have ρ critical races (r_1, v_1), (r_2, v_2), …, (r_ρ, v_ρ), and let $T(r_i, v_i) := \{r_i^1, r_i^2, \ldots, r_i^{m(i)}\} \subset X$ be the set of all outcomes of the critical race (r_i, v_i), $i = 1, \ldots, \rho$. Divide $T(r_i, v_i)$ into subsets $C_i^1, C_i^2, \ldots, C_i^{m(i)}$ according to the output value of $r_i^1, r_i^2, \ldots, r_i^{m(i)}$. These subsets $C_i^1, C_i^2, \ldots, C_i^{m(i)}$ can be represented by $x_1, x_2, \ldots, x_c$, which are called the common-output states of the machine Σ_g. Set $m = n + t + c$, then the generalized state set increases to $\tilde{X} = \{x^1, \ldots, x^m\}$.♦

Note that the outcomes of the critical races may be a combination of burst states and regular states. Moreover, if two subsets contain the same states, then they are represented by the same common-output state.

After introducing the common-output state of the machine Σ_g, we should update the generalized one-step reachability matrix $R(\Sigma_g)$ and the generalized reachability matrix $\Gamma(\Sigma_g)$. The transitions from one generalized state to a subset of states, which have the same output value, will be replaced by the single transition from the starting state to a newly defined common-output state.

Example 3-13 Consider a machine Σ with the input alphabet A = {a, b, c}, the output alphabet Y = {0, 1, 2}, and the state set $X = \{x^1, x^2, x^3, x^4, x^5\}$. There is a critical race pair (x^1, c) in the machine Σ (Table 3-2).

Table 3-2. Transition table of the machine Σ

	a	b	c	Y
x^1	x^1	x^4	$\{x^4, x^5\}$	0
x^2	x^2	x^1	x^2	1
x^3	x^1	x^5	x^3	1
x^4	x^1	x^4	x^2	2
x^5	x^2	x^5	x^3	2

The stable state machine of Σ is $\Sigma_{|s}$ (Table 3-3).

Table 3-3. Transition table of the machine $\Sigma_{|s}$

	a	b	c	Y
x^1	x^1	x^4	$\{x^2, x^3\}$	0
x^2	x^2	x^4	x^2	1
x^3	x^1	x^5	x^3	1
x^4	x^1	x^4	x^2	2
x^5	x^2	x^5	x^3	2

Using Algorithm 2-13, we get the generalized stable recursion function s_g of the machine Σ_g. Associate a burst state x^6 with the subset $\{x^2, x^3\}$. Then the generalized machine Σ_g has a generalized state set $\tilde{X} = \{x^1, ..., x^6\}$. Assign a common-output state x^7 to represent the subset

$\{x^4, x^5\}$. Now the generalized machine Σ_g has a generalized state set $\tilde{X} = \{x^1, ..., x^m\}$ and $m =$ 7. The generalized one-step reachability matrix is

$$R(\Sigma_g) = \begin{bmatrix} a & N & N & b & N & c & N \\ N & \{a,c\} & N & b & N & N & N \\ a & N & c & N & b & N & N \\ a & c & N & b & N & N & N \\ N & a & c & N & b & N & N \\ a & a & N & N & N & c & b \\ a & a & N & N & N & c & b \end{bmatrix}. \blacklozenge$$

Definition 3-14 Let $\Sigma_{|s}$ be a generalized stable state machine with the generalized stable recursion function s_g, and let u be an input string of $\Sigma_{|s.}$ The transition induced by u from a generalized state x is a deterministic transition if $s_g(x, u)$ consists of a single state. The machine Σ_g is a deterministic machine if all transitions of $\Sigma_{|s}$ are deterministic. ♦

3.3 Output Feedback Trajectory

From the definition of the critical race, we know a transition from a critical race pair to the outcomes is not a deterministic transition. So, originally, an asynchronous machine with critical races is not a deterministic machine. If we can transform the machine with critical races into a deterministic machine on a specific basis, then we actually get rid of the effect of the critical races to the machine. The following procedure helps us to transform a machine with critical races into a deterministic machine.

Consider a generalized machine Σ_g with the generalized state set $\{x^1, x^2, ..., x^m\}$ and input set A. Assume the machine has a critical race (x^j, v) with the outcomes $\{x^p, x^q\}$ and $p \neq$

q. If the $h(x^p) = h(x^q)$, then we can define a burst state or a common-output state to make the transition from x^j to the subset $\{x^p, x^q\}$ a deterministic transition. So, we only need to focus on the situation that the outcomes have different output values, i.e., $h(x^p) \neq h(x^q)$. In another word, we cannot make the generalized machine Σ_g transform into a deterministic one simply with a generalized state set. When $h(x^p) \neq h(x^q)$, if there exist input strings which can take the machine Σ_g from these two states (also two output values) to a single generalized state x^s through deterministic transitions, then the effect of the critical race (x^j, v) can be eliminated. Assume there exist input strings $u^1, u^2 \in A^+$, where u^1 takes the machine from x^p to x^s deterministically and u^2 takes the machine from x^q to x^s deterministically. That means $s_g(x^p, u^1) = s_g(x^q, u^2) = x^s$. Then we can generate a deterministic transition from x^j to x^s with introducing an output feedback controller to the machine Σ_g as follows: After applying the input v at the state x^j, check the outcomes. If the outcome is x^p, then apply u^1 to the machine. If the outcome is x^q, then apply u^2 to the machine. Hence, based on the generalized state set of a generalized machine Σ_g, we can turn the machine Σ_g into a deterministic machine with an output feedback controller. This controller sets up a standard projection on the generalized state set $\tilde{X}$, which can be denoted by $\Pi_x : \tilde{X} \times A \rightarrow \tilde{X} : \Pi_x(x, u) = x$.

Definition 3-15 Let Σ_g be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, ..., x^m\}$, the input alphabet A, and the generalized stable recursion function s_g. An output feedback trajectory from the generalized state x^j to the common-output state x^i is a list $\{S_0, S_1, ..., S_p\}$ of sets of valid pairs of $\Sigma_{|s}$ with the following properties:

(i) $s_g(x, u)$ is a detectable transition for all $(x,u) \in U_{j=0,...,p} S_j$,

(ii) $S_0 = \{(x^j, u_0)\}$,

(iii) $s_g\,[S_\alpha] \subset \Pi_x[S_{\alpha+1}]$, $\alpha = 0, ..., p-1$,

(iv) $s_g\,[S_p] = \{x^i\}$. ♦

For example, consider the machine Σ_g in Example 3-13. The output feedback trajectory from the generalized state x^6 to the generalized state x^4 is: $S_0 = \{(x^6, a)\}$, $S_1 = \{(x^1, b), (x^2, b)\}$.

Proposition 3-16 Let Σ_g be a generalized machine and let x^j and x^i be two generalized states of Σ_g. The following two statements are equivalent.

(a) There exists an output feedback controller C that drives Σ_g through a deterministic transition from x^j to x^i.

(b) There is an output feedback trajectory from x^j to x^i.

Proof. Suppose that the part (b) is valid, and let $\{S_0, S_1, \ldots, S_p\}$ be an output feedback trajectory from x^j to x^i. We construct an output feedback controller C that takes the machine Σ_g from x^j to x^i through a string of deterministic transitions. This output feedback controller C has two inputs: one is the output burst $\beta \in Y^*$ of Σ_g, and the other is the external input $v \in A$, which is also the command input of the controller C. Given a set of characters $W \in A$, we need to construct a controller $C(x^j, x^i, W)$. This controller takes the machine Σ_g from generalized state x^j to x^i through deterministic transitions as the response to an input character $w \in W$.

The controller $C(x^j, x^i, W)$ is a combination of an observer B and a control unit F as shown in Figure 2-3. Where, the observer is an input/state machine $B = (A \times Y^*, \tilde{X}, Z, z_0, \sigma, I)$ with two inputs: the input character $u \in A$ of Σ_g and the output burst $\beta \in Y^*$ of Σ_g; the state set Z is identical to the generalized state set $\tilde{X}$, and the initial condition is identical to that of

Σ_g, i.e., $z_0 = x_0$. And the function $\sigma : Z \times A \times Y^* \rightarrow Z$ is the stable recursion function of B. Let z_k be the state of the observer B at the step k, while ω_k be the output of B. The observer B is an input/state machine defined by the recursion (Eq. 2-4)

The control unit F is also an input/state asynchronous machine $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ with two inputs: the external input $v \in A$ and the output $\omega \in \tilde{X}$ of the observer B. To complete the construction of the controller C, we need to derive the recursion function ϕ and the output function η of the unit F.

According to Figure 2-3, as long as $v \notin W$, the controller C stays in its initial state (z_0, ξ_0) and the input character u of the machine Σ_g equals to the external input character v. After the machine Σ_g arrives a stable combination of state x^j, if the external input v changes to a character of W, then the controller C starts working. The observer B collects both the input character u and the output burst $\beta(x^j)$ of the machine Σ_g and feeds the control unit F with the state of the machine Σ_g. The control unit F generates a string of characters $u_1u_2\ldots u_r$ and apply it to the machine Σ_g. This input string $u_1u_2\ldots u_r$ will drive the machine Σ_g from the state x^j to x^i through a string of detectable and stable deterministic transitions.

Recalling that the control unit is an input/state asynchronous machine $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$. The recursion function of F is a function $\phi : \Xi \times \tilde{X} \times A \rightarrow \Xi$ and the output function of F is denoted by $\eta : \Xi \times \tilde{X} \times A \rightarrow A$. Referring to Figure 2-3, the output $\omega \in \tilde{X}$ of the observer B is one of the inputs of F, and the other input is the external input $v \in W$. Then the control unit F generate the string $u \in A$ to feed the controlled machine Σ_g. Note that the control unit F must

operates in a fundamental mode, so the whole system must have reached a stable combination before the F generates the next input character for Σ_g. Assume that F will generate r input characters $u_1u_2...u_r$ to feed Σ_g, then it needs r states $\xi^1(x^j, w), \xi^2(x^j, w), ..., \xi^r(x^j, w)$. Denote this set by $\Xi(x^j, w):=\{\xi^1(x^j, w), \xi^2(x^j, w), ..., \xi^r(x^j, w)\}$. We define the recursion function ϕ and output function η of the F as follows.

(i) Let $U(x^j)\subset A$ be the set of all input characters that form stable combination with the generalized state x^j, and let z_0 be the initial state of B and ξ_0 be the initial state of F. Set

$$\phi(\xi_0, (z, t)) := \xi_0 \text{ for all } (z, t)\in \tilde{X}\times A\backslash x^j\times U(x^j),$$

$$\phi(\xi_0, (x^j, v)) := \xi_1(x^j) \text{ for all } v\in U(x^j).$$

Where $\xi_1(x^j)$ is the state of F, when observer B detects a stable combination with x^j.

When both B and F are at initial states, the controller $C(x^j, x^i, W)$ directly applies the external input v to Σ_g, thus set

$$\eta(\xi_0, (z, v)) :=v \text{ for all } (z, v)\in \tilde{X}\times A.$$

(ii) When the observer B detects a stable combination of Σ_g with the generalized state x^j, suppose the external input switches to a character $w\in W$. We choose a character $u_j\in U(x^j)$ and set

$$\eta(\xi(x^j), (x^j, t)) := u_j \text{ for all } t\in U(x^j).$$

In this way, the machine Σ_g lingers in the state x^j when the external input switches to a character of W. Hence, the fundamental mode operation of the machine is guaranteed. Then the control unit F will generate an input string $u=u_1u_2...u_r$ to drive the machine Σ_g to the

generalized state x^i. Since we have a output feedback trajectory $\{S_0, S_1, ..., S_p\}$, we need P new states for F, where

$$P = \#\Pi_x S_0 + \#\Pi_x S_1 + ... + \#\Pi_x S_p. \tag{3–7}$$

Denote the state of F as $\xi^k(x^j, w, x)$, where $x \in \Pi_x S_k$ and $k = 1, ..., p$. When the input character switches to w, the control unit F moves to the state $\xi^0(x^j, w, x^j)$ and it begins to generate the first input character u_0 to feed Σ_g, where $u_0 \in A$ is an character that satisfies $(x^j, u_0) \in S_0$. To implement this, we set

$$\phi(\xi_1(x^j), x^j, w) := \xi^0(x^j, w, x^j) \text{ for all } w \in W;$$

$$\phi(\xi_1(x^j), x^j, v) := \xi_1(x^j) \text{ for all } v \in U(x^j) \setminus W;$$

$$\phi(\xi_1(x^j), x^j, v) := \xi_0 \text{ for all } v \notin U(x^j) \cup W;$$

$$\eta(\xi^0(x^j, w, x^j), x, v) := u_0 \text{ for all } (x, v) \in X \times A.$$

After u_0 is applied to Σ_g, the machine will move to a generalized stable combination with a generalized state $x_1 \in \Pi_x S_1$. After the observer B detects this transition, the control unit F moves to the next state $\xi^1(x^j, w, x_1)$ and generates the next input character u_1 to the machine Σ_g. The process continues similarly until x^i is reached. Then at the step $k \in \{1, 2, ..., p\}$ the function ϕ and η must be defined as follows

$$\phi(\xi^{k-1}(x^j, w, x_{k-1}), x_k, w) := \xi^k(x^j, w, x_k),$$

$$\eta(\xi^k(x^j, w, x_k), x, v) := u_k \text{ for all } (x, v) \in X \times A.$$

At the k=p step, the machine Σ_g reaches a state $x_p \in \Pi_x S_p$. Set $s_g(x_p, u_p) := x^i$, where u_p is an character satisfies $(x_p, u_p) \in S_p$. This is accomplished by setting

$$\phi(\xi^p(x^j, w, x^i), x_k, w) := \xi^{p+1}(x^j, w, x^i),$$

$$\phi(\xi^{p+1}(x^j, w, x^i), z, t) := \xi_0 \text{ for all } (z, t) \in X \times A \backslash W.$$

$$\eta(\xi^{p+1}(x^j, w, x^i), x, v) := u_p \text{ for all } (x, v) \in X \times A.$$

As long as the external input remains as a character in set W, the machine will linger in the stable combination (x^i, u_p). If the external input is no longer belongs to W, the controller returns to the initial state ξ_0. We build the controller as well as prove that statement (b) implies statement (a) as above.

Conversely, assume part (a) is valid. Let ξ_0 be the initial state of the controller C and let $C(\xi, x, u)$ be the output value produced by the controller C when it is at the next stable state corresponding to its state ξ, Σ_g is at the state x and the external input is u. By assumption, there is an external input value w that induces the controller C to generate an input string $u_0u_1...u_p$ for the machine Σ_g from the generalized state x^j to the generalized state x^i via deterministic transition. The first character of this input string is $u_0 = C(\xi_0, x^j, u)$. Define the set $S_0 := \{(x^j, u_0)\}$.

Let s_g be the generalized stable recursion function of Σ_g. when the input from the controller to Σ_g changes to u_0, Σ_g moves to a generalized stable combination with one of the states of the set $s(x^j, u_0) = s[S_0]$. When this state is reached by Σ_g, the controller C detects the new state and controller moves to its own next stable state. Let $\xi(x, u_0)$ be the next stable state of the controller and let $u_1 = C(\xi(x, u_0), x, w) \in A$ be the output character generated by the controller once C reaches $\xi(x, u_0)$. Define the set

$$S_1 := \{(x, C(\xi(x, u_0), x, w)) : x \in s(S_0)\}.$$

Continue operating like that until the set S_k, $k > 0$, is defined. Build a new set by setting

$$S_{k+1} := \{(x', C(\xi(x', u_k), x', w)) : x' \in s(S_k)\}.$$

By assumption, the controller C drives Σ_g to the state x^i through deterministic transitions. Consequently, there exists an integer p such that $s(S_p) = x^i$. Then, the list $S_0, S_1, ..., S_p$ forms a output feedback trajectory. So, the existence of a output feedback controller C that drive Σ_g from x^j to x^i through deterministic transitions, implies the existence of a output feedback trajectory $S_0, S_1, ..., S_p$ from x^j to x^i. Namely, part (a) implies part (b). This completes the proof. ♦

3.4 Preliminary Generalized Skeleton Matrix

Using the algorithm in the above proof of Proposition 3-16, we can check the basic connection between any two generalized states of the machine Σ_g, namely, the existence of the output feedback trajectory between any pair of generalized states. If we focus on the stable and detectable reachability properties, then we don't need to record all the input strings. Instead, we can use a numerical matrix, which has only entries of one and zero, to represent this one-step stable and detectable reachability properties. This numerical matrix can be called "preliminary generalized skeleton matrix" of the machine Σ_g. Afterwards, it is easier to calculate the power of it and obtain the "overall preliminary generalized skeleton matrix". For the machine Σ_g with m generalized states, the overall preliminary generalized skeleton matrix characters all the stable and detectable transitions among the generalized states of the machine within m steps. We can use the following algorithm to gradually transform the generalized one-step reachability matrix into the preliminary one-step generalized skeleton matrix. Meanwhile, the machine Σ_g is transformed into a deterministic machine on the generalized state basis with an output feedback controller C.

An operation involving strings of A^+ and zero and 1 should be defined before giving the algorithm.

Definition 3-17 Let ω be a character not included in A. The meet operation between two strings of A^+ and zero and 1 is defined as follow:

$0 \wedge 0 := 0, 0 \wedge 1 = 1 \wedge 0 := 0, 1 \wedge 1 := 1,$

$0 \wedge a = a \wedge 0 := 0, 1 \wedge a = a \wedge 1 := \omega,$ for all $a \in A^+$.

The meet of two vectors with $r \geq 1$ components is defined entrywise as the vector of the meets of the corresponding components. ♦

Algorithm 3-18 Let Σ_g be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, ..., x^m\}$ and let $\Gamma(\Sigma_g)$ be the generalized stable reachability matrix of the generalized machine Σ_g.

Step 1: Transpose the matrix $\Gamma(\Sigma_g)$ and denote the resulting matrix by $\Gamma'(\Sigma_g)$.

Step 2: Replace all entries of N in the matrix $\Gamma'(\Sigma_g)$ by the number 0; denote the resulting matrix by K^1.

Step 3: Perform (a) below for each $i,j = 1, ..., m$; then continue to (b):

(a) If K^1_{ij} includes a string of A^+ that does not appear in any other entry of the same column j, then replace entry K^1_{ij} by the number 1. Otherwise, let the entry K^1_{ij} remain.

(b) Set $k := 1$ and denote the resulting matrix by $K(k)$.

Step 4: If all entries of row k of the matrix $K(k)$ are 1 or 0, then set $K(k+1) := K(k)$ and set $k := k+1$.

Step 5: If $k = m + 1$, then set $K_1(\Sigma_g) := K(k)$ and terminate the algorithm. Otherwise, go to step 6.

Step 6: Perform the following operations:

(a)If there is a character $u \in A$ that appears in row k of $K(k)$, then let $j_1, ..., j_q$ be the columns of row k of $K(k)$ that include u. Denote by $J(u)$ the meet of rows $j_1, ..., j_q$ of the matrix $K(k)$.

(b) If $J(u)$ has no entries other than 0 or 1, then delete u from all entries of row k of the matrix $K(k)$; set all empty entries, if any, to the value 0. Continue to (c).

(c) If $J(u)$ has no entries of 1, then return to Step 3. Otherwise, continue to (d).

(d) If $J(u)$ has entries of 1, then let $j_1, ..., j_r$ be the entries of $J(u)$ having the value 1. Let $S(k)$ be the set of rows of $K(k)$ that consists of row k and of every row that has the number 1 in row k. In the matrix $K(k)$, perform the following operations on every row of $S(k)$:

Delete from the column all occurrences of input characters that appear in columns $j_1, j_2, ..., j_r$ of the row.

Replace rows $j_1, j_2, ..., j_r$ of the column by the number 1.

If any entries of $K(k)$ remain empty, then replace them by the number 0. Return to Step 4.

The final resulting matrix $K_1(\Sigma_g)$ is called the preliminary generalized skeleton matrix of the machine Σ_g. ♦

Definition 3-19 The outcome $K_1(\Sigma_g)$ of Algorithm 3.23 is defined as the preliminary generalized skeleton matrix of the generalized asynchronous machine Σ_g.♦

Note that this preliminary generalized skeleton matrix $K_1(\Sigma_g)$ of the generalized machine Σ_g is similar to the one-step fused skeleton matrix $\Delta(\Sigma)$ of an asynchronous machine Σ in Geng and Hammer (2005).

Example 3-20 Consider the machine Σ and the generalized machine Σ_g of Example 3-10. The generalized stable reachability matrix $\Gamma(\Sigma_g)$ of the machine Σ_g is

$$\Gamma(\Sigma g) = \begin{bmatrix} \begin{Bmatrix} a,ab,ac,aba,aca \\ abab,abac,acab,acac \end{Bmatrix} & N & N & \begin{Bmatrix} b,ba,bab,bac \\ baba,baca \end{Bmatrix} & \begin{Bmatrix} c,ca,cab,cac \\ caba,caca \end{Bmatrix} \\ \begin{Bmatrix} a,ab,ac,aba,abc \\ aca,abab,abac \end{Bmatrix} & c & N & \begin{Bmatrix} b,ba,bc,bab,bac \\ baba,babc,baca \end{Bmatrix} & \begin{Bmatrix} ca,cab,cac,caba \\ cabc,caca \end{Bmatrix} \\ \begin{Bmatrix} a,ac,aba,aca \\ abac,acac \end{Bmatrix} & N & c & \{ba,bc,bac,baca,baba\} & \{ca,cac,caba,caca\} \\ \begin{Bmatrix} a,ab,aba,aca \\ abab,acab \end{Bmatrix} & N & N & \{b,ba,bab,baba,baca\} & \{ca,cab,caba,caca\} \\ \begin{Bmatrix} a,ac,aba,aca \\ abac,acac \end{Bmatrix} & N & N & \{ba,bac,baba,baca\} & \{c,ca,cac,caba,caca\} \end{bmatrix}.$$

Applying the Algorithm 3-18 to the matrix $\Gamma(\Sigma_g)$, we can obtain the preliminary generalized skeleton matrix $K_1(\Sigma_g)$ of the machine Σ_g is

$$K_1(\Sigma_g) = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}. \blacklozenge$$

Proposition 3-21 Let Σ_g be a generalized machine with the preliminary generalized skeleton matrix $K_1(\Sigma_g)$, and let x^i and x^j be two generalized states of Σ_g. Then the following two statements are equivalent.

(a) There exists an output feedback trajectory from x^j to x^i.

(b) The (i, j) entry of $K_1(\Sigma_g)$ is 1.♦

CHAPTER 4
MODEL MATCHING FOR INPUT/OUTPUT ASYNCHRONOUS MACHINES WITH RACES

In the present chapter we start to address the model-matching problem for input/output asynchronous machines with critical races. In last chapter the reachability properties of an input/output asynchronous machine with critical races has been discussed and corresponding generalized machine has been derived. The newly defined generalized machine with related generalized state set and generalized functions of the machine could be controlled as a deterministic machine without critical races. The control of this kind of asynchronous machines has been discussed in Geng and Hammer (2005). Thus, the controller will be designed to correct the input/output machine under the configuration of Fig. 2-3, so that the closed-loop system possesses an equivalent input/output behavior as that of a prescribed model.

Since we are discussing the input/output machines, we first study the equivalent list of the generalized machine Σ_g, with respect to the model Σ'. Then we work on the sufficient and necessary conditions of the existence of the output feedback controllers so as to solve the model matching problem. When such a controller exists, we provide an algorithm to construct the controller. Finally, an example is presented to illustrate how the control system operates.

4.1 Model Matching Problem

As we mentioned before, the design of a controller to eliminate the effects of critical races of an existing asynchronous machine is called the Model-Matching Problem. Specifically, the formal statement of the model matching problem is as follows. Let Σ be a machine that exhibits undesirable behavior. Assume that the desirable behavior is specified by an asynchronous machine Σ'. The machine Σ' is called the model. Our objective is to design a controller C for which the behavior of the closed loop system Σ_c simulates the behavior of the model Σ'. It is

indicated in Kohavi (1970) that the practical performance of an asynchronous machine is determined by its stable-state behavior. Thus, the stable-state behavior of Σ_c need to be equivalent to the stable-state behavior of Σ'. Let us first introduce the classical notions of equivalence.

Definition 4-1 Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be two machines having the same input and the same output sets, and let $\Sigma_{|s}$ and $\Sigma'_{|s}$ be the stable state machines induced by Σ and Σ', respectively. Two states $x \in X$ and $\zeta \in X'$ are stably equivalent ($x \equiv \zeta$) if the following conditions are true: When $\Sigma_{|s}$ starts from the state x and $\Sigma'_{|s}$ starts from the state ζ, then (i) $\Sigma_{|s}$ and $\Sigma'_{|s}$ have the same permissible input strings; (ii) $\Sigma_{|s}$ and $\Sigma'_{|s}$ generate the same output string for every permissible input string. The two machines Σ and Σ' are stably equivalent if their initial states are stably equivalent, i.e., if $x_0 \equiv \zeta_0$. ♦

Note that two machines $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ that are stably equivalent appear identical to a user.

Definition 4-2 Given a machine Σ and Σ', find necessary and sufficient conditions for the existence of a controller C such that Σ_c is stably equivalent to Σ' and operates in fundamental mode. If such a controller C exists, derive an algorithm for its design. ♦

In this dissertation, the model matching problem concentrates on matching the stable input/output behavior of the model. The model Σ' can be taken as a stably minimal machine. Let Σ_g be a generalized machine with the generalized state set $\{x^1, x^2, ..., x^m\}$ which is induced from the machine Σ. Our objective is then to match the input/output behavior of the generalized machine Σ_g and the model Σ'.

Next, let us introduce a notion which underlies the solution of the model matching problem for asynchronous machines. Given two sets S^1 and S^2 and a function $g : S^1 \to S^2$, denote by g^I the inverse set function of g; i.e., for an element $s \in S^2$, the value $g^I(s)$ is the set of all elements $\alpha \in S^1$ that satisfies $g(\alpha) = s$.

Definition 4-3 Let $\Sigma = (A, Y, X, s, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be two machines having the same input and the same output sets. Let $\Sigma_g = (A, Y, \tilde{X}, s_g, h_g)$ be a generalized machine induced from the machine Σ. The state set X' of Σ' consist of the q state $\zeta^1, ..., \zeta^q$. Define the subsets $E^i := h_g^I h'(\zeta^i) \subset X$, $i = 1, ..., q$. Then, $E(\Sigma_g, \Sigma') := \{E^1, ..., E^q\}$ is the output equivalence list of Σ_g with respect to Σ'. ♦

An equivalence list is characterized by the following property: the value of the output function h_g of Σ_g at any state of the set E^i is equal to the value of the output function h' of Σ' at the state ζ^i. The members of an output equivalence list are not necessarily disjoint sets.

Definition 4-4 Let Σ_g be a generalized machine with generalized state set $\tilde{X} = \{x^1, ..., x^m\}$, and let Λ^1 and Λ^2 be two nonempty subsets of $\tilde{X}$. The reachability indicator $r(\Sigma_g, \Lambda^1, \Lambda^2)$ is defined as 1 if every element of Λ^1 can reach an element of Λ^2 through a chain of stable and detectable transitions; otherwise, $r(\Sigma_g, \Lambda^1, \Lambda^2) = 0$. ♦

Example 4-5 Let Σ_g be a generalized machine with generalized state set $\tilde{X} = \{x^1, x^2, x^3\}$ and the preliminary generalized skeleton matrix is $K_1(\Sigma_g)$,

$$K_1(\Sigma_g) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Let $\Lambda^1 = \{x^1, x^2\}$ and $\Lambda^2 = \{x^2, x^3\}$ be two state subsets. Then

$$r(\Sigma_g, \Lambda^1, \Lambda^2) = 1. \blacklozenge$$

Definition 4-6 Let Σ_g be a generalized machine with generalized state set $\tilde{X} = \{x^1, ..., x^m\}$, and let $\Lambda = \{\Lambda^1, ..., \Lambda^q\}$ be a list of $m \geq 1$ nonempty subsets of $\tilde{X}$. The fused skeleton matrix $\Delta(\Sigma_g, \Lambda)$ of Λ is an $q \times q$ matrix whose (i,j) entry is

$$\Delta_{ij}(\Sigma_g, \Lambda) = r(\Sigma_g, \Lambda^i, \Lambda^j). \blacklozenge$$

Example 4-7 Consider the machine Σ_g and the two state subsets Λ^1 and Λ^2 in the Example 4-5. Let $\Lambda := \{\Lambda^1, \Lambda^2\}$ be a list of subsets of $\tilde{X}$. Then the fused skeleton matrix $\Delta(\Sigma_g, \Lambda)$ of Λ is $\Delta(\Sigma_g, \Lambda)$.

$$\Delta(\Sigma_g, \Lambda) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \blacklozenge$$

Definition 4-8 Let $\Lambda = \{\Lambda^1, ..., \Lambda^q\}$ and $W=\{W^1, ..., W^q\}$ be two lists of subsets of $\tilde{X}$. The length of the list Λ is the number q of its members. The list W is a subordinate list of the list Λ, denoted as $W \prec \Lambda$, if it has the same length q as the list Λ and if $W^i \subset \Lambda^i$ for all $i=1, ..., m$. A list is deficient if it includes the empty set $\varnothing$ as one of its members. $\blacklozenge$

4.2 Existence of Controllers

Next, we give the condition of the existence of a controller C for which Σ_c is stably equivalent to a specified model Σ'. Given two $p \times q$ numerical matrices A and B, the expression $A \geq B$ indicates that every entry of the matrix A is not less than the corresponding entry of the matrix B, i.e., $A_{ij} \geq B_{ij}$ for all $i = 1,..., p$ and for all $j = 1, ..., q$.

Lemma 4-9 Let $\Sigma_g=(A, Y, \tilde{X}, x_0, s_g, h_g)$ and $\Sigma'=(A, Y, X', s', h')$ be asynchronous machines, where Σ' is stably minimal. Let $X'= \{\zeta^1, ..., \zeta^q\}$ be the state set of Σ', where the initial condition of Σ' is $\zeta_0 = \zeta^d$. Assume that there is a controller C for which Σ_c is stably equivalent to Σ' and operates in fundamental mode. Then, there is a non-deficient subordinate list Λ of the output equivalence list $E(\Sigma_g, \Sigma')$ for which $\Delta(\Sigma_g, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$. ♦

The proof of the above Lemma 4-9 is similar to the proof of the Lemma 4.11 in Geng and Hammer (2005). The difference is that a generalized machine appears here instead of a regular machine Σ. Recall that all the underlying states of a burst state or a common output state in the generalized machine Σ_g are the same states in the original machine Σ and those underlying states have the same output value. Thus, when applying a real input character u to the generalized machine Σ_g at a generalized state x', it is the same to apply this input character u to the real machine Σ at any underlying state of that generalized state x'. The real machine Σ will generates the same output value as the generalized machine Σ_g does.

The condition of Lemma 4-9 is not only a necessary condition, but also a sufficient condition for the existence of a controller to solve the model matching problem. The inequality $\Delta(\Sigma_g, \Lambda) \geq K(\Sigma')$ guarantees that the corresponding output values of the two machines match. If the model Σ' has a stable transition from a state ζ^i to state ζ^j, then the machine Σ_g has a

stable and detectable transition from every state in Λ^i to a state in Λ^j. Thus, we only need to construct a controller C which generates the input string that takes Σ_g from a state in Λ^i to a state in Λ^j. This controller should be a combination of an observer and a control unit as described in Figure 2-3.

Theorem 4-10 Let $\Sigma_g=(A, Y, \tilde{X}, x_0, s_g, h_g)$ and $\Sigma'=(A, Y, X', s', h')$ be stably reachable asynchronous machines, where Σ' is stably minimal. . Let $X'= \{\zeta^1, ..., \zeta^q\}$ be the state set of Σ', where the initial condition of Σ' is $\zeta_0 = \zeta^d$. Then the following two statements are equivalent.

(i) There is a controller C for which $\Sigma_c = \Sigma'$, where Σ_c operates in fundamental mode and is well posed.

(ii) There is a non-deficient subordinate list Λ of the output equivalence list $E(\Sigma_g, \Sigma')$ such that $\Delta(\Sigma_g, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.

Moreover, when (ii) holds, the controller C can be designed as a combination of an observer B and a control unit F as depicted in Figure 2-3 and the observer is given by Equation 2-4.♦

Proof. The generalized machine Σ_g has a generalized state set $\tilde{X} = \{x^1, x^2, ..., x^t\}$. All the underlying states of the generalized states $\{x^1, x^2, ..., x^t\}$ in this set $\tilde{X}$ are the same states in the set X of the original machine Σ. Since all the real states included in a burst state or in a common output state will work with the same input value, the same input value can be used on the real machine. Furthermore, the output of a burst state or a common output state is the same as that of the underlying states. Hence, the operation of the real machine is as same as before the

introducing of the generalized machine. Thus, we can use the same method in Geng and Hammer (2005) on the generalized machine, i.e., to find a controller for Σ_g.

The Lemma 4-9 indicates that statement (i) implies statement (ii). Now let us assume that (ii) is valid. Let $\Lambda=\{\Lambda^1, ..., \Lambda^q\}$ be a subordinate list of $E(\Sigma_g, \Sigma')$ satisfying $\Delta(\Sigma_g, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$. Using Λ, we build a controller C for which the closed loop system Σ_c of 1.1 is stably equivalent to the model Σ', is well posed, and operates in fundamental mode. The controller C is a combination of an observer B and a control unit F as depicted in Figure 2-3. The observer B is given by Proposition 3-16, so we complete the proof by constructing the control unit F. Recall that the control unit F is an asynchronous machine $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ with two inputs: the external input $v \in A$ and the output $\omega \in \tilde{X}$ of the observer B. To complete the construction of the controller C, we need to derive the recursion function ϕ and the output function η of the unit F.

Assume that Σ' is at the stable state ζ^i and that Σ_g is at a stable state $\chi \in \Lambda^i$. Note that ζ^i is either the initial condition $x_0 \in \Lambda^d$ of Σ' or the outcome of a detectable stable transition; χ is either the initial condition $x_0 \in \Lambda^d$ of Σ_g or the outcome of a detectable stable transition. Assume the external input character switches to the character w. Then the model Σ' moves to its next stable state $s'(\zeta^i, w)=\zeta^j$. Recall that s_g is the generalized stable recursion function of Σ_g. The inequality $\Delta(\Sigma_g, \Lambda) \geq K(\Sigma')$ implies that there is an input string $u=u_1u_2...u_r$ such that the stable combinations $(\chi, u_1), (s_g(\chi, u_1), u_2), \ldots, (s_g(\chi, u_1u_2\ldots u_{r-1}), u_r)$ are all detectable, and such that the state $x_r := s_g(\chi, u)$ belongs to Λ^j. Define the intermediate states

$$X_1 := s_g(\chi, u_1), x_2 := s_g(x_1, u_2), \ldots, x_r = s_g(x_{r-1}, u_r). \tag{4-1}$$

As the combinations (x_i, u_i), i= 1, …, r, are all stable and detectable combinations, the states $x_1, \ldots, x_r$ appear as output values of the observer B immediately after having been reached by Σ_g. The situation can be depicted as follows.

$$\begin{array}{lcl} \Sigma': & \zeta^i & \xrightarrow{\quad w \quad} \zeta^j \\ \Sigma_g: & \chi \in \Lambda^i & \xrightarrow{\quad u_1u_2\ldots u_r \quad} x_r \in \Lambda^j \end{array}$$

Figure 4-1. Equivalence of two asynchronous machines Σ' and Σ_g

The objective of the control unit F is to generate the string $u = u_1u_2\ldots u_r$ and apply it as input to the real machine Σ. This action achieves model matching for the present transition for the following reason. The string u drives the system Σ_g to the stable state x_r, which then becomes the next stable state of the closed loop system Σ_c. Then, since $h(x_r) = h[\Lambda^j] = h'(\zeta^j)$, the next stable state of Σ_c produces the same output value as the model Σ' to match the model's response.

Note that the control unit F must operates in a fundamental mode, so the whole system must have reached a stable combination before the F generates the next input character for Σ_g. Then, we construct a recursion function ϕ for F to implement the above behavior. Keeping in mind the requirement of fundamental mode operation of the machines, we need to make sure that the control unit F generates the string u one character at a time and at each step that the composite system has reached a stable combination before generating the next character. As the string u has r characters, the control unit F needs r states to accomplish this: $\xi^1(\chi, \zeta^i, w), \ldots, \xi^r(\chi, \zeta^i, w)$. The resulting set of states

$$\Xi(\chi, \zeta^i, w) := \{\xi^1(\chi, \zeta^i, w), \ldots, \xi^r(\chi, \zeta^i, w)\}$$

is associated with the state ζ^i of Σ', the state χ of Σ, and the external input character w. To account for all possible such combinations, the control unit F needs the state set

$$\Xi := \xi_0 \cup \{\cup_{i-1}, \ldots, \cup^{\chi \in \Lambda\, i}_{w \in A} \Xi(\chi, \zeta^i, \omega)\},$$

where ξ_0 is the initial state of F. We shall use the following notation. For a state x of the machine Σ_g, let

$$U(x) := \{a \in A : s_g(x, a) = x\}$$

be the set of all input characters that form stable combinations with x. Similarly, for a state ζ of the machine Σ', let

$$U'(\zeta) := \{a \in A : s'(\zeta, a) = \zeta\}$$

be the set of all input characters that form stable combinations with ζ.

Recalling that the control unit is an input/state asynchronous machine $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$. The recursion function of F is a function $\phi: \Xi \times \tilde{X} \times A \to \Xi$ and the output function of F is denoted by $\eta : \Xi \times \tilde{X} \times A \to A$. Referring to the configuration (2.24), the output $\omega \in \tilde{X}$ of the observer B is one of the inputs of F, and the other input is the external input $v \in W$. Then, ϕ and η are defined as follows.

(i) Let the closed loop system Σ_c be at a stable combination, where Σ_g is at the state χ, namely, the real machine Σ is at one of the underlying states of the generalized state χ. The observer B has he output value $\omega = \chi$, and control unit F is at a state $\xi \in \Xi$. Select an element $c \in U(\chi)$, and define

$$\phi(\xi, (\chi, b)) := \xi \text{ for all } b \in U'(\zeta^i),$$

$$\eta(\xi,(\chi,a)) := c \quad \text{for all} \quad a \in A.$$

This guaranties that the closed loop system Σ_c and the model Σ' operate in fundamental mode.

(ii) Suppose that the external input switches to a character w satisfying $s'(\zeta^i, w) = \zeta^j$. Then, the control unit F needs to generate the input string $u = u_1u_2\ldots u_r$, to take Σ_g through the chain of states $x_1, \ldots, x_r$ to the state $x_r \in \Lambda^j$. Meanwhile the output of the observer B will track the state sequence $x_1, \ldots, x_r$. Thus, the recursion function ϕ is defined as follows.

$$\phi(\xi,(\chi, w)) := \xi^1(\chi, \zeta^i, w),$$

$$\phi(\xi^k(\chi, \zeta^i, w),(x_k, w)) := \xi^{k+1}(\chi, \zeta^i, w), \qquad k = 1, 2, \ldots, r\text{-}1,$$

$$\eta(\xi^k(\chi, \zeta^i, w),(z, b)) := u_k, \qquad \text{for any } (z,b) \in \tilde{X}\times A,\ k = 1, 2, \ldots, r.$$

(iii) In response to the last input character u_r produced by F, the machine Σ_g reaches the desired stable state x_r, which implies the real machine Σ reaches one of the underlying states of the generalized stable state x_r. The machine Σ_g needs to remain at the state x_r until the external input switches from w to another character. Then, choose an element $v \in U(x_r)$ and assign

$$\phi(\xi^r(\chi, \zeta^i, w),(x_r, w)) := \xi^r(\chi, \zeta^i, w),$$

$$\eta(\xi^r(\chi, \zeta^i, w),(z, b)) := v, \qquad \text{for all } (z,b) \in \tilde{X}\times A.$$

This completes the construction of the control unit F. Note that whenever the machine Σ_g is at a generalized state x, the real machine Σ is at an corresponding underlying state x' of this generalized state x and $h_g(x) = h(x')$. This construction achieves model matching of the generalized machine Σ_g to the model Σ' with fundamental model operation as well. This concludes the proof. ♦

The proof of Theorem 4-10 includes an algorithm for the construction of a controller C solving the model matching problem. Then, we use the Algorithm 4.14 in Geng and Hammer (2005) to build a list Λ that satisfies condition (ii) of this theorem whenever such a list exists. This algorithm and Theorem 4-10 give a comprehensive and constructive solution of the model matching problem. A recursive process is used in the algorithm to build a decreasing chain of subordinate lists. The last list in this chain, if not deficient, satisfies condition (ii) of Theorem 4-10; if the last list of the chain is deficient, then there is no controller that solves the requisite model matching problem.

Let $\Sigma_g=(A, Y, \tilde{X}, x_0, s_g, h_g)$ and $\Sigma'=(A, Y, X', s', h')$ be the machines of Theorem 4.8, let $E(\Sigma_g, \Sigma') = \{E^1, ..., E^q\}$ be their output equivalence list, and let $K(\Sigma')$ be the skeleton matrix of Σ'. The following steps yield a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ ... \succ \Lambda(r)$ of subordinate lists of $E(\Sigma_g, \Sigma')$. The members of the list $\Lambda(i)$ are denoted by $\Lambda^1(i), ..., \Lambda^q(i)$; they are subsets of the state set $\tilde{X}$ of Σ_g.

Algorithm 4-11 Let $\Sigma_g=(A, Y, \tilde{X}, x_0, s_g, h_g)$ and $\Sigma'=(A, Y, X', s', h')$ be the machines of Theorem 4-10, let $E(\Sigma_g, \Sigma') = \{E^1, ..., E^q\}$ be their output equivalence list, and let $K(\Sigma')$ be the skeleton matrix of Σ'. The following steps yield a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ ... \succ \Lambda(r)$ of subordinate lists of $E(\Sigma_g, \Sigma')$. The members of the list $\Lambda(i)$ are denoted by $\Lambda^1(i), ..., \Lambda^q(i)$; they are subsets of the state set $\tilde{X}$ of Σ_g.

Start Step: Set $\Lambda(0) := E(\Sigma_g, \Sigma')$.

Recursion Step: Assume that a subordinate list $\Lambda(k) = \{\Lambda^1(k), ..., \Lambda^q(k)\}$ of $E(\Sigma_g, \Sigma')$ has been constructed for some integer $k \geq 0$. For each pair of integers $i,j \in \{1, ..., q\}$, let $S_{ij}(k)$ be

the set of all states $x \in \Lambda^i(k)$ for which the (i,j) element of $\Delta(\Sigma_g, \Lambda^j(k))$ is 0; i.e., $S_{ij}(k)$ consists of all states $x \in \Lambda^i(k)$ for which there is no chain of stable and detectable transitions to a state of $\Lambda^j(k)$. Note that $S_{ij}(k)$ may be empty. Then set

$$T_{ij}(k) := \begin{cases} S_{ij}(k) & \text{if } K_{ij}(\Sigma')=1; \\ \phi & \text{if } K_{ij}(\Sigma')=0. \end{cases}$$

Now, using \ to denote set difference, define the subsets

$$V^i(k) := \cup_{j=1,\ldots,q} T_{ij}(k),\ i = 1, \ldots, q$$

$$\Lambda^i(k+1) := \Lambda^i(k) \backslash V^i(k),\ i = 1, \ldots, q$$

Then, the next subordinate list in our decreasing chain is given by

$$\Lambda(k+1) := \{ \Lambda^1(k+1), \ldots, \Lambda^q(k+1)\}.$$

Test Step: the algorithm terminates if the list $\Lambda(k+1)$ is deficient or if $\Lambda(k+1) = \Lambda(k)$; otherwise, repeat the Recursion Step, replacing k by $k+1$. ♦

4.3 A Comprehensive Example of Controller Design

Consider an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ with the input alphabet A={a, b, c}, the output alphabet Y={0, 1, 2}, and the state set $X=\{x^1, x^2, x^3, x^4\}$. There is a critical race pair (x^1, c) in the machine (Tables 2-1, Figure 2-1). Let another machine $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be the desired model (Table 4-1, Figure 4-2).

The initial state of Σ is $x_0 = x^1$ and the initial state of Σ' is $\zeta_0 = \zeta^1$. After introducing a burst state $x^5 = \{x^2, x^3\}$, we have the generalized machine Σ_g of Σ (Table 4-2).

Table 4-1. Transition table of the machine Σ'

	a	b	c	Y
ζ^1	ζ^1	ζ^2	ζ^3	0
ζ^2	ζ^1	ζ^2	-	1
ζ^3	ζ^3	-	ζ^3	2

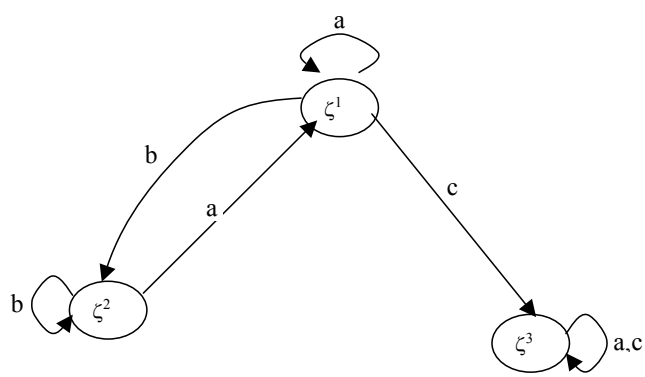

Figure 4-2. State flow diagram of the machine Σ'

Table 4-2. Stable state transition table of the machine Σ_g

	a	b	c	Y
x^1	x^1	x^4	x^5	0
x^2	x^1	x^4	x^2	1
x^3	x^1	-	x^3	1
x^4	x^1	x^4	-	2
x^5	x^1	-	x^5	1

The initial state of Σ_g is $x_0 = x^1$. From Table 2-1 and Table 4-2., the output equivalence list is $E(\Sigma_g, \Sigma') = \{E^1, E^2, E^3\}$, where $E^1=\{x^1\}$, $E^2=\{x^2, x^3, x^5\}$, $E^3=\{x^4\}$. The preliminary generalized skeleton matrix of the generalized machine is $K_1(\Sigma_g)$ and the skeleton matrix of the model is $K(\Sigma')$.

$$K_1(\Sigma_g) = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$K(\Sigma') = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

The subordinate list Λ of the output equivalence list $E(\Sigma_g, \Sigma')$ is

$\Lambda^1(1) = \{x^1\}$, $\Lambda^2(1) = \{x^2, x^3, x^5\}$, $\Lambda^3(1) = \{x^4\}$.

The fused skeleton matrix is $\Delta(\Sigma_g, \Lambda)$ that satisfies

$$\Delta(\Sigma_g, \Lambda) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \geq \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = K(\Sigma') \ .$$

Thus, there exists a controller C to turn the machine Σ into a deterministic machine that matches the model Σ'.

We have a generalized machine $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ with the state set $\tilde{X} = \{x^1, \ldots, x^5\}$ and a subordinate list $\Lambda(1) = \{\Lambda^1(1), \Lambda^2(1), \Lambda^3(1)\}$ that satisfies $\Delta(\Sigma_g, \Lambda(1)) \geq K(\Sigma')$ and $x^1 \in \Lambda^1(1)$. According to the process of construction of the controller C that is described before, we derive the control unit F and combine it with the observer B of Equation 2-4. Then, we have the corrective controller C as shown in Figure 2-3.

Recall that the initial state of Σ_g is $x_0 = x^1$ and the initial state of Σ' is $\zeta_0 = \zeta^1$. Maintain the external input character as a to keep Σ' at the state ζ^1 and maintain the external input character as a to keep Σ_g at the state x^1. Now we construct F as follows. Set the initial state of F to ξ_0. Denote the states of the observer B by $\{x^1, x^2, x^3, x^4, x^5\}$ and set the initial state of B to x^1. Thus, we have $U(x^1) = \{a\}$ and $U'(\zeta^1) = \{a\}$ and

F: $\phi(\xi_0, (x^1, a)) := \xi_0,$

$\eta(\xi_0, (x^1, a)) := a.$

B: $\sigma(x^1, (a, \beta)) = x^1$ for all $\beta \in Y^*$.

Assume then the external input character switches from a to b. For the machine Σ', $s'(\zeta^1, b) = \zeta^2$ and $h'(\zeta^1) = 0$ and $h'(\zeta^2) = 1$, so this transition is detectable. In order to simulate this transition, the system Σ_g has to move to a state in $\Lambda^2 = \{x^2, x^3, x^5\}$. Since $s_g(x^1, c) = x^5$. For the real machine Σ, it needs to move to either state x^2 or state x^3 and it does not matter in which state the Σ really stays. In either case, the control unit F needs to generate the character c to serve as input for Σ so that

F: $\phi(\xi_0, (x^1, b)) = \xi^1(x^1, \zeta^1, b),$

$\eta(\xi^1(x^1, \zeta^1, b), (x^1, b)) = c.$

B: $\sigma(x^1, (c, \beta)) = \begin{cases} x^5 & \text{for } \beta = 01; \\ x^1 & \text{otherwise.} \end{cases}$

F: $\phi(\xi^1(x^1, \zeta^1, b), (x^5, b)) = \xi^1(x^1, \zeta^1, b),$

$\eta(\xi^1(x^1, \zeta^1, b), (x^5, b)) = c.$

B: $\sigma(x^5, (c, \beta)) = x^5$ for all $\beta \in Y^*$.

Now consider the other option: the machine Σ_g is at a stable combination with the state $x^1 \in \Lambda^1$ and Σ' is at a stable combination with the state ζ^1, when the external input character switches from a to c. For the machine Σ', $s'(\zeta^1, c) = \zeta^3$ and $h'(\zeta^1) = 0$ and $h'(\zeta^3) = 2$. Thus this transition is detectable as well. To simulate this transition, the machine Σ_g needs to move to a state in $\Lambda^3=\{x^4\}$, i.e. to x^4 . So does the real machine Σ. Since $s(x^1, b) = x^4$, the control unit F needs to generate the character b and this leads to the following

F: $\phi(\xi_0, (x^1, c)) = \xi^1(x^1, \zeta^1, c)$,

$\eta(\xi^1(x^1, \zeta^1, c), (x^1, c)) = b$.

Σ_g: $s(x^1, b) = x^4$,

$\beta(x^1, b) = h(x^1)h(x^4) = 02$

B: $\sigma(x^1, (b, \beta)) = \begin{cases} x^4 & \text{for } \beta = 02; \\ x^1 & \text{otherwise.} \end{cases}$

Then, assume the machine Σ stays at a stable combination with the state $x^2 \in \Lambda^2$ and the model is at a stable combination with the state ζ^2, when the external input character switches to a. The model's response is $s'(\zeta^2, a) = \zeta^1$ so F needs to generate an input character a to drive Σ to a state in $\Lambda^1 =\{x^1\}$. So we have

F: $\phi(\xi^1(x^1, \zeta^1, b), (x^2, a)) = \xi^1(x^2, \zeta^2, a)$,

$\eta(\xi^1(x^2, \zeta^2, a), (x^2, a)) = a$.

Σ_g: $s(x^2, a) = x^1$,

$\beta(x^2, a) = h(x^2)h(x^1) = 10$

B: $\sigma(x^2, (a, \beta)) = \begin{cases} x^1 & \text{for } \beta = 10; \\ x^2 & \text{otherwise.} \end{cases}$

Another possibility is that Σ is in a stable combination with the state $x^3 \in \Lambda^2$ and the model is at a stable combination with the state ζ^2, when the external input character switches from b to a. The model's response is $s'(\zeta^2, a) = \zeta^1$ so F needs to generate an input character a to drive Σ to a state in $\Lambda^1 = \{x^1\}$. Then

F: $\phi(\xi^1(x^1, \zeta^1, b), (x^3, a)) = \xi^1(x^3, \zeta^2, a)$,

$\eta(\xi^1(x^3, \zeta^2, a), (x^3, a)) = a$.

Σ_g: $s(x^3, a) = x^1$,

$\beta(x^3, a) = h(x^3)h(x^1) = 10$

B: $\sigma(x^3, (a, \beta)) = \begin{cases} x^1 & \text{for } \beta = 10; \\ x^3 & \text{otherwise.} \end{cases}$

Assume further that Σ_g is in a stable combination with the state $x^5 \in \Lambda^2$ and the model is at a stable combination with the state ζ^2, when the external input character switches from b to a. This case is a combination of the above two cases. Thus, F will generate an input a to drive Σ to a state in $\Lambda^1 = \{x^1\}$.

F: $\phi(\xi^1(x^1, \zeta^1, b), (x^5, a)) = \xi^1(x^5, \zeta^2, a)$,

$\eta(\xi^1(x^5, \zeta^2, a), (x^5, a)) = a$.

Σ_g: $s(x^5, a) = x^1$,

$\beta(x^5, a) = h(x^3)h(x^1) = 10$

B: $\sigma(x^5, (a, \beta)) = \begin{cases} x^1 & \text{for } \beta = 10; \\ x^5 & \text{otherwise.} \end{cases}$

This completes the construction of the corrective controller C for the model matching problem. The state set of the control unit F is

$$\Xi = \{\xi_0, \xi^1(x^1, \zeta^1, b), \xi^1(x^1, \zeta^1, c), \xi^1(x^2, \zeta^2, a), \xi^1(x^3, \zeta^2, a), \xi^1(x^5, \zeta^2, a)\}.$$

The above state set can be reduced to three states and F can be depicted by Figure 4-3 (the notation near the arrows indicates the information of (output of the observer B, input of Σ_c)/output of F).

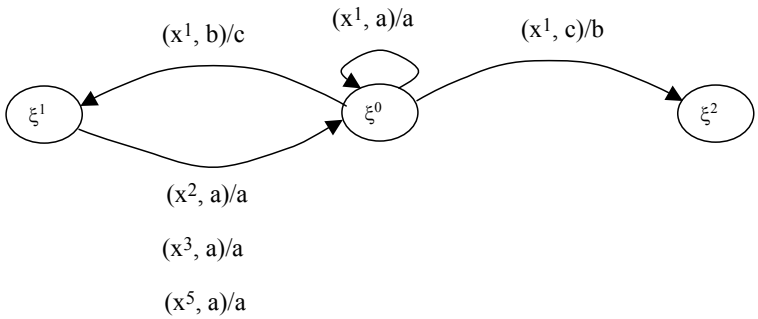

Figure 4-3. State transitions diagram of control unit F

The observer $B = (A \times Y^*, \tilde{X}, Z, z_0, \sigma, I)$ is defined according to Equation 2-4 and described by Figure 4-4. The controller is the combination of F and B according to Figure 2-3.

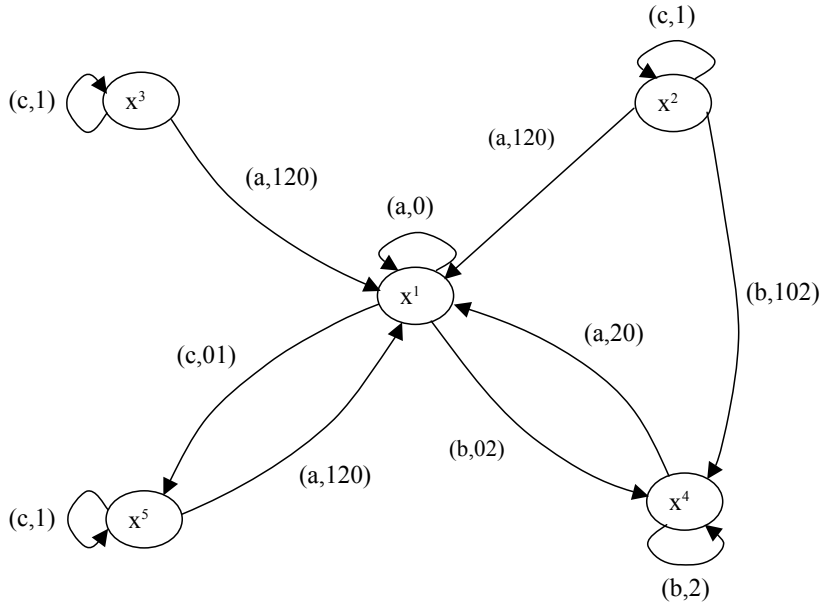

Figure 4-4. State transitions diagram of observer B

CHAPTER 5
SUMMARY AND FUTURE WORK

In the present work, the feedback controllers are introduced to correct the faulty behavior of asynchronous machines. When critical races afflict the asynchronous machine, the existence of the controllers offer a solution to eliminate the effects of the critical races while controlling the machine to match a desirable race-free model. We call the problems as Model-Matching Problems. This approach discloses an interesting and constructive field in which many related topics are worth investigating.

The solutions have been obtained to the Model-Matching Problem for asynchronous input/output machines. The concept of generalized state has been used to describe a persistent state of the machine Σ about which only partial information is available. The generalized state allows us to use the partial information available about the state of Σ to continue controlling the machine as best as possible toward the goal of achieving model matching, while taking best advantage of the available information about Σ. The results of the Model-Matching Problem include necessary and sufficient conditions for the existence of the controller, and algorithms for its construction whenever a controller exists.

The following list is the possible topics for future research:

(i) The algorithm for transforming the generalized stable reachability matrix into skeleton matrix has been proposed in chapter 3. However, before that we need to raise the power of the generalized one-step reachability matrix, which requires a large amount of computation. When the state set of the machine is large, this issue is more significant. If we can obtain a likely one-step skeleton matrix from the one-step reachability matrix and raise the power of this numerical skeleton matrix instead, then the calculation is much simpler. But we need spend time to keep all the information that we need in the transforming and computation.

(ii) The introduction of generalized state transforms an asynchronous machine with critical races into a deterministic machine. However, the state space is enlarged depending on the number of critical races. If we can minimize the state space, then it will increase the speed of computation significantly too.

(iii) Although we can construct an output feedback controller for the closed loop system to eliminate the effects of critical race whenever a controller exists, this controller may not be minimal. We can also work on this issue to find out a good strategy to minimize the controller.

(iv) The present discussion excludes the existence of infinite cycles in the existing machine. We shall deal with the situation when both critical races and infinite cycles occur in the defective machine.

The controller constructed in the present work ensures that the closed-loop system in Figure 1-1 and Figure 2-3 operates in fundamental mode. The input changes were only allowed during stable combinations. This requires the restriction of the controlled machine to those without any unstable cycles; otherwise the controller can't do anything to correct the machine once the machine enters a cycle.

LIST OF REFERENCES

Alpan, G. and Jafari, M.A., ″Synthesis of a closed-loop combined plant and controller model,″ *IEEE Trans. on Systems, Man and Cybernetics*, vol. 32, no. 2, 2002, pp. 163-175.

Barrett, G., and Lafortune, S., ″Bisimulation, the supervisory control problem, and strong model matching for finite state machines,″ *Journal of Discrete Event Dynamic Systems*, Volume 8, number 4, 1998, pages 377-429.

Chu, T., ″Synthesis of hazard-free control circuits from asynchronous finite state machines specifications,″ *Journal of VLSI Signal Processing*, vol.7, no.1-2, 1994, p. 61-84.

Datta, P.K., Bandyopadhyay, S.K. and Choudhury, A.K., "A graph theoretic approach for state assignment of asynchronous sequential machines," *International Journal of Electronics*, vol.65, no.6, 1988, p. 1067-75.

Davis, A., Coates, B. and Stevens, K., ″The post office experience: designing a large asynchronous chip,″ *Proceeding of the Twenty-Sixth Hawaii International Conference on System Sciences*, vol.1, pp. 409-418.

Dibenedetto, M.D., ″Nonlinear strong model matching,″ *IEEE Trans. Automatic Control*, vol. 39, 1994, pp. 1351-1355.

Dibenedetto, M.D., Saldanha, A., and Sangiovanni-Vincentelli, A., ″Model matching for finite state machines, ″ *Proceedings of the IEEE Conf. on Decision and Control*, vol. 3, 1994, pp. 3117-3124.

Dibenedetto, M.D., Saldanha, A., and Sangiovanni-Vincentelli, A., ″Strong model matching for finite state machines with nondeterministic reference model,″ *Proceedings of the IEEE Conf. on Decision and Control*, vol. 1, 1995, pp. 422-426.

Dibenedetto, M.D., Saldanha, A., and Sangiovanni-Vincentelli, A., ″Strong model matching for finite state machines, ″ *Proceedings of European Control Conference*, vol. 3, 1995, pp. 2027-2034.

Dibenedetto, M.D., Sangiovanni-Vincentelli, A., and Villa, T., ″Model matching for finite-state machines,″ *IEEE Transactions on Automatic Control*, vol. 46, no. 11, 2001, pp. 1726-1743.

Eilenberg, S., ″Automata, Languages, and Machines,″ Academic Press, NY, 1994.

Furber, S.B., ″Breaking step the return of asynchronous logic,″ *IEE Review*, 1993, pp. 159-162.

Fisher, P.D., and Wu S. F., ″Race-free state assignments for synthesizing large-scale asynchronous sequential logic circuits,″ *IEEE Transactions on Computers*, vol. 42, no. 9, 1993, pp. 1025-1034.

Geng, X., ″Model matching for asynchronous sequential machines,″ Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA, 2003

Geng, X., Hammer, J., ″Input/output control of asynchronous sequential machines, ″ *IEEE Trans. on Automatic Control*, Vol. 50, No. 12, pp 1956-1970.

Hammer, J., ″On some control problems in molecular biology, ″ *Proceedings of the IEEE Conference on Decision and Control*, Vol. 4, 1994, pp. 4098-4103.

Hammer, J., ″On the modeling and control of biological signaling chains, ″ *Proceedings of the IEEE Conference on Decision and Control*, Vol. 4, 1995, pp. 3747-3752.

Hammer, J., ″On corrective control of sequential machines, ″ *International Journal of Control*, Vol. 65, No. 65, 1996, pp. 249-276.

Hammer, J., ″On the control of incompletely described sequential machines, ″ *International Journal of Control* Vol. 63, No. 6, 1996, pp. 1005-1028.

Hauck, S., ″Asynchronous design methodologies: an overview,″ *Proceedings of the IEEE*, vol. 83, no. 1, 1995, pp. 69-93.

Higham, L. and Schenk, E., ″The parallel asynchronous recursion model,″ *Proceedings of the IEEE Symposium on Parallel and Distributed Processing*, 1992, 310–316.

Holcombe, W.M.L, ″Algebraic Automata Theory, ″ *Cambridge University Press*, New York, 1982.

Hubbard, P. and Caines, P.E., ″Dynamical consistency in hierarchical supervisory control,″ *IEEE Trans. on Automatic Control*, vol. 47, no. 1, 2002, pp. 37-52.

Huffman, D.A., [1954a] ″The synthesis of sequential switching circuits, ″ *J. Franklin Inst.*, vol. 257, pp. 161-190.

Huffman, D.A., ″The synthesis of sequential switching circuits, ″ *J. Franklin Inst*., vol. 257, 1954, pp. 275-303.

Huffman, D.A., ″The design and use of hazard-free switching networks, ″ *Journal of the Association of Computing Machinery*, vol. 4, no. 1, 1957, pp. 47-62.

Isidori, A., ″The matching of a prescribed linear input-output behavior in a nonlinear system,″ *IEEE Trans. Automatic Control*, vol. AC-30, 1985, pp. 258-265.

Kam, T., Villa, T., Brayton, R., and Sangiovanni-Vincentelli, A., ″Synthesis of FSMs : Functional Optimization, ″ Boston, MA: Kluwer, 1997.

Kam, T., Villa, T., Brayton, R., and Sangiovanni-Vincentelli, A., ″Implicit computation of compatible sets for state minimization of ISFSMs,″ *IEEE Transactions on Computer Aided Design*, vol. 16, 1997, pp. 657-676.

Kam, T., Villa, T., Brayton, R., and Sangiovanni-Vincentelli, A., ″Theory and algorithms for state minimization of nondeterministic FSMs,″ *IEEE Transactions on Computer Aided Design*, vol. 16, 1997, pp. 1311-1322.

Kohavi, Z., ″Switching and Finite Automata Theory, ″ McGraw-Hill Book Company, New York, 1970.

Koutsoukos, X.D., Antsaklis, P.J., Stiver, J.A. and Lemmon, M.D., ″Supervisory control of hybrid systems,″ *Proceedings of the IEEE*, vol. 88, no. 7, 2000, pp. 1026-1049.

Lavagno, L., Keutzer, K., and Sangicivanni-Vincentelli, A., ″Algorithms for synthesis of hazard-free asynchronous circuits,″ *Proceedings of the 28th ACM/IEEE Conference on Design Automation*, 1991, pp. 302–308.

Lavagno, L., Moon, C. W., and Sangiovanni-Vincentelli, A., ″Efficient heuristic procedure for solving the state assignment problem for event-based specifications, ″ *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, 1994, pp 45-60.

Lin, B. and Devadas, S., ″Synthesis of hazard-free multilevel logic under multiple-input changes from binary decision diagrams,″ *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 8, 1995, pp. 974-985.

Lin, F., ″Robust and adaptive supervisory control of discrete event systems, ″ *IEEE Transactions on Automatic Control*, vol. 38, n0. 12, 1993, pp. 1848-1852.

Maki, G., and Tracey, J., ″A state assignment procedure for asynchronous sequential circuits, ″ *IEEE Transactions on Computers*, vol. 20, 1971, pp. 666-668.

Marshall, A., Coates, B. and Siegel, F., ″Designing an asynchronous communications chip,″ *IEEE Design & Test of Computers*, Vol. 11, no. 2, 1994, pp. 8-21.

Mealy, G.H., ″A method for synthesizing sequential circuits,″ *Bell System Tech. J.*, vol. 34, 1955, pp. 1045-1079.

Moon, C.W., Stephan, P.R., and Brayton, R.K., ″ Synthesis of hazard-free asynchronous circuits from graphical specifications,″ *IEEE International Conference on Computer-Aided Design*, 1991, pp. 322 –325.

Moore, B. and Silverman, L., ″Model matching by state feedback and dynamic compensation,″ *IEEE Trans. Automatic Control*, vol. 17, 1972, pp. 491-497.

Moore, E.F., ″Gedanken-experiments on sequential machines,″ *Automata Studies, Annals of Mathematical Studies*, no. 34, Princeton University Press, N.J., 1956.

Moore, S.W., Taylor, G.S., Cunningham, P.A., Mullins, R.D. and Robinson, P., ″Self-calibrating clocks for globally asynchronous locally synchronous systems,″ *Proceedings of Inter. Confer. on Computer Design*, 2000, pp. 73-78.

Morse, A.S., ″Structure and design of linear model following systems″, *IEEE Trans. Automatic Control*, vol. 18, 1973, pp. 346-354.

Murphy T.E., Geng X., and Hammer J., ″Controlling races in asynchronous sequential machines,″ *Proceeding of the IFAC World Congress*, Barcelona, July 2002..

Murphy T.E., Geng X., and Hammer J., ″On the control of asynchronous machines with races,″ *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1073-1081.

Murphy, T.E., ″On the control of asynchronous sequential machines with races, ″ Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA, 1996.

Nishimura, N., ″Efficient asynchronous simulation of a class of synchronous parallel algorithms,″ *Journal of Computer and System Sciences*, vol. 50, no. 1, 1995, pp. 98-113.

Nowick, S.M., ″Automatic synthesis of burst-mode asynchronous controllers,″ Ph.D. Dissertation, Stanford University, 1993.

Nowick, S.M., and Coates, B., ″UCLOCK: automated design of high-performance unclocked state machines,″ *ICCD '94. Proceedings., IEEE International Conference on Computer Design*, 1994, pp. 434–441.

Nowick, S.M., Dean, M.E., Dill, D.L. and Horowitz, M., ″The design of a high-performance cache controller: a case study in asynchronous synthesis,″ *Proceedings of the 26th Hawaii International Conference on System Sciences*, 1993, pp. 419–427.

Nowick, S.M. and Dill, D.L., ″Synthesis of asynchronous state machines using a local clock,″ *IEEE International Conference on Computer Design*, 1991, pp. 192-197.

Oliveira, D.L., Strum, M., Wang, J.C. and Cunha, W.C., ″Synthesis of high performance extended burst mode asynchronous state machines,″ *Proceedings. 13th Symposium on Integrated Circuits and Systems Design*, 2000, pp. 41-46

Ozveren, C.M., Willsky, A.S., and Antsaklis, P.J., ″Stability and stabilizability of discrete event systems, ″ *J. ACM*, Vol. 38, 1991, pp. 730-752.

Park, S.-J. and Lim, J.-T., ″Robust and nonblocking supervisory control of nondeterministic discrete event systems using trajectory models, ″ *IEEE Trans. on Automatic Control*, vol. 47, no. 4, 2002, pp. 655-658.

Peterson, J.L., ″Petri Net Theory and The Modeling of Systems, ″ Prentice-Hall, NJ, 1981.

Ramadge, P.J.G., and Wonham, W.M., ″supervisory control of a class of discrete event processes,″ *SIAM Journal of Control and Optimization*, vol. 25, no. 1, 1987, pp. 206–230.

Ramadge, P.J.G., and Wonham, W.M., ″The control of discrete event systems,″ *Proceedings of IEEE*, vol. 77, no. 1, 1989, pp. 81-98.

Cole, R. and Zajicek, O., ″The expected advantage of asynchrony,″ *Journal of Computer and System Science*, vol. 51, no. 2, pp. 286-300.

Shields, M.W., ″An Introduction to Automata Theory, ″ Blackwell Scientific Publications, Boston, 1987.

Thistle, J. G. and Wonham, W.M., ″Control of infinite behavior of finite automata, ″ *SIAM Journal on Control and Optimization*, vol. 32, no. 4, 1994, pp 1075-1097.

Unger, S. H., ″Asynchronous Sequential Switching Circuits, ″ Wiley-Interscience, New York, NY, 1969.

Unger, S. H., ″Self-synchronizing circuits and non-fundamental mode operation,″ *IEEE Trans. Computers*, vol. 26, no. 3, 1977, pp. 278-281.

Unger, S. H., ″Hazards, critical races, and metastability, ″ *IEEE Trans. on Computers*, vol. 44, no. 6, 1995, pp 754-768.

Venkatraman, N., ″On the control of asynchronous sequential machines with infinite cycles, ″ Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA, 2004

Venkatraman, N., ″On the control of asynchronous sequential machines with infinite cycles, ″ *International Journal of Control*, Vol. 79, No. 07, 2006, pp. 764-785.

Yu, M.L. and Subrahmanyam, P.A., ″A path-oriented approach for reducing hazards in asynchronous designs,″ *Proceedings of Design Automation Conference 29th ACM/IEEE* , 1992, pp. 239 –244.

BIOGRAPHICAL SKETCH

Jun Peng was born in Wuhan, Hubei Province, China. She received her bachelor's degree in automatic control and master's degree in control theory and control engineering from Shanghai Jiao Tong University, Shanghai, China, in July 2000 and in April 2003, respectively. She began her Ph.D. program in the Department of Electrical and Computer Engineering at University of Florida, Gainesville, FL in August 2003. Her research interests include asynchronous sequential circuits, application of asynchronous sequential systems in computer architecture, artificial intelligence and biological systems, control theory, control systems, and applications of control theory in computer communication networks. She received her Ph.D. in August 2007.