Proceeding of the European Control Conference, Budapest, Hungary, August 2009

Generalized Realizations and Output Feedback Control of Asynchronous Sequential Machines with Races

Jun Peng and Jacob Hammer

***Abstract*— The novel notion of a 'generalized realization' for asynchronous sequential machines with critical races is presented as a tool for the design of output feedback controllers that eliminate the uncertainty caused by critical races. A generalized realization helps represent the uncertainty induced by critical races and creates a deterministic relationship between input-output data and the generalized state. This makes it possible to use deterministic techniques to design output feedback controllers for non-deterministic asynchronous sequential machines.**

I. INTRODUCTION

Asynchronous sequential machines serve as building blocks of a wide variety of engineering systems, including high speed computing systems, parallel computing systems, and models of processes in molecular biology (e.g., HAMMER [1994]). This note deals with the development of control techniques that help overcome an important potential defect of asynchronous sequential machines - the presence of critical races. Critical races cause a machine to exhibit unpredictable behavior; they may originate from malfunctions, design flaws, implementation flaws, or genetic flaws in biological systems. Usually, the presence of critical races makes it impossible to determine the exact state of an asynchronous machine from input-output data.

The notion of a 'generalized realization', introduced in section III, facilitates a simple methodology for the design of output feedback controllers that eliminate the effects of critical races. A generalized realization helps represent the uncertainty induced by critical races in a way that creates a deterministic relationship between input-output data and the generalized state of a machine. This allows us to adapt output feedback control techniques of deterministic machines for use in non-deterministic cases. The output feedback configuration of interest is as follows.

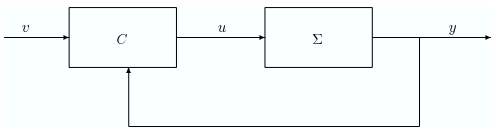


Fig. 1. A Feedback Control Configuration

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

The desirable behavior of Σ_c is represented by a deterministic asynchronous machine Σ', called a *model*. We seek an output feedback controller C for which Σ_c emulates Σ'. As Σ' is deterministic, this eliminates the effects of uncertainties caused by critical races of Σ.

Recall that an asynchronous machine may occupy a *stable state* or a *transient state*. A stable state is a state at which the machine lingers until a change occurs in one of its input variables. A transient state is a state through which the machine passes very quickly, ideally in zero time. An asynchronous machine may pass through several transient states on its way from one stable state to another. In principle, the controller C operates by turning undesirable stable states of Σ into transient states of the closed loop machine Σ_c.

To avoid adding uncertainty to an asynchronous machine's behavior, care must be taken to keep its input constant while the machine is in transition. Otherwise, input changes may occur at unpredictable states during transition and result in an unpredictable outcome. In this regard, asynchronous machines are normally operated in *fundamental mode*, where input changes are allowed only while a machine is in a stable state. All asynchronous machines in this note are operated in fundamental mode.

The string of outputs that an asynchronous machine generates on its way from one stable state to the next is called a *burst* - a rapidly progressing string of output characters. Often, control objectives can be achieved without utilizing bursts. The present note concentrates on controllers that do not utilize bursts, as this often leads to simpler controllers.

Other aspects of the control of asynchronous machines are discussed in MURPHY, GENG, and HAMMER [2002 and 2003], VENKATRAMAN and HAMMER [2004], GENG and HAMMER [2005], and YANG and HAMMER [2007a, b]. The design of output feedback controllers for asynchronous machines with critical races requires the development of new theoretical notions, such as the notion of generalized realization considered in this note.

Studies dealing with additional facets of the control of sequential machines can be found in RAMADGE and WONHAM [1987] and THISTLE and WONHAM [1994], where the theory of discrete event systems is investigated; in HAMMER [1994, 1995, 1996a and b, 1997], DIBENEDETTO, SALDANHA, and SANGIOVANNI-VINCENTELLI [1994], and BARRETT and LAFORTUNE [1998]), where issues related to control and model matching for sequential ma-

Jun Peng and Jacob Hammer are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA. email: hammer@mst.ufl.edu

chines are considered. These discussions do not take into consideration specialized issues related to the operation of asynchronous machines, such as stable states, transient states, and fundamental mode operation.

The paper is organized as follows. Section II explores requirements for fundamental mode operation of asynchronous machines with critical races; section III introduces generalized realizations; section IV covers estimation of generalized states from input-output data; and section V provides a perspective on the control configuration we employ. The paper concludes in section VI with a discussion of the reachability features of generalized realizations and their applications to controller design.

II. Strong Detectability

An asynchronous sequential machine is a sextuple $\Sigma = (A, Y, X, x_0, f, h)$, where A is the *input alphabet*, Y is the *output alphabet*, X is the *state set*, $x_0 \in X$ is the initial state of the machine, $f : AX \to X$ is the *recursion function*, and $h : X \to Y$ is the *output function*; all involved sets are finite. The machine operates according to

$$\Sigma : \begin{cases} x_{k+1} = f(x_k, u_k) \\ y_k = h(x_k), k = 0, 1, 2, ..., \end{cases} \quad (1)$$

where $x_k \in X$ is the state, $u_k \in A$ is the input value, and $y_k \in Y$ is the output value at step k. The step counter k advances by one upon a change of the input or of the state of Σ. Iterations of the recursion function with constant input u are defined by

$$f^i(x, u) := f(f^{i-1}(x, u), u), f^0(x, u) := x, i = 1, 2, ...$$

A *valid pair* $(x, u) \in X \times A$ is a pair at which the recursion function f is defined; (x, u) is a *stable combination* if $x = f(x, u)$. The machine Σ lingers at a stable combination until the input character u is changed. When (x, u) is not a stable combination, the machine generates a chain of transitions

$$x_1 = f(x, u), x_2 = f(x_1, u), ... \quad (2)$$

which may or may not terminate. If this chain of transitions terminates, the last state, say x_q, satisfies $x_q = f(x_q, u)$; then, (x_q, u) is a stable combination and x_q is *the next stable state* of x with the input u. If the chain (2) does not terminate, then it forms an *infinite cycle*. In this paper, we consider only asynchronous machines with no infinite cycles.

A *critical race* is a valid pair $(r, v) \in X \times A$ for which the next stable state is not uniquely determined, but may be one of several *outcomes*, say $r_1, r_2, ..., r_m \in X$ (e.g., KOHAVI [1970]). An asynchronous machine with critical races is nondeterministic. Its recursion function f may have sets of states as its values, and the symbols x_{k+1} in (1) and $x_1, x_2, ...$ in (2) may represent sets of states.

State transitions of asynchronous machines occur very quickly - ideally, in zero time. As a result, the observed behavior of a machine is determined by its stable combinations. Specifically, for a valid pair (x, u), let x' the next stable state or, if (x, u) is a critical race, let $r_1, r_2, ..., r_m$ be the outcomes. Then, the *stable recursion function* s of Σ is

$$s(x, u) := \begin{cases} x' \text{ if } (x, u) \text{ is not a critical race,} \\ r_1, r_2, ..., r_m \text{ if } (x, u) \text{ is a critical race.} \end{cases}$$

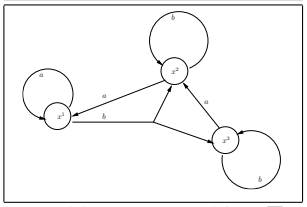
The *stable state machine* $\Sigma_{|s} = (A, Y, X, x_0, s, h)$ describes the stable transitions of Σ. Two asynchronous machines Σ and Σ' are *stably equivalent* if the stable state machines $\Sigma_{|s}$ and $\Sigma'_{|s}$ have the same input/output behavior. Stably equivalent machines are indistinguishable by a user, so we write $\Sigma = \Sigma'$ when Σ and Σ' are stably equivalent.

Example 2.1: Consider a machine $\Sigma = (A, Y, X, x_0, f, h)$ with $A = \{a, b\}$, $Y = \{0, 1\}$, $X = \{x_1, x_2, x_3\}$, and the transitions:

Σ: Transition Table

X	a	b	Y
x^1	x^1	$\{x^2, x^3\}$	0
x^2	x^1	x^2	1
x^3	x^2	x^3	1

Σ: Transition Diagram

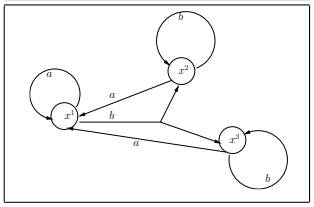


Here, (x^1, b) is a critical race. The stable state machine $\Sigma_{|s}$ has the transitions:

$\Sigma_{|s}$: Transition Table

X	a	b	Y
x^1	x^1	$\{x^2, x^3\}$	0
x^2	x^1	x^2	1
x^3	x^1	x^3	1

$\Sigma_{|s}$: Transition Diagram



To guarantee that no uncertainties are introduced by improper operation, our asynchronous machines are operated in fundamental mode: inputs are changed only while the machine is in a stable combination (e.g., KOHAVI [1970]). Then, a machine is in a well defined state when an input change occurs. Fundamental mode operation is the most common mode of operating asynchronous machines.

Condition 2.2: The configuration of Figure 1 operates in fundamental mode when all the following hold.

(i) C is in a stable combination while Σ undergoes transitions;

(ii) Σ is in a stable combination while C undergoes transitions; and

(iii) The external input v changes only while Σ and C are both in a stable combination. □

To implement Condition 2.2*(i)*, it must be possible for the controller C to determine from input and output values of Σ whether Σ has completed its transitions and reached the next stable state. This leads to the following notion.

Definition 2.3: Assume that the machine Σ is at a stable combination with the state x, when the input character switches to u. The pair (x, u) is *strongly detectable* if it can be determined from input and output values of Σ whether the next stable state has been reached.

Let S be a set of states of Σ. Assume that Σ is at a stable combination with an unspecified state $x \in S$, when the input character switches to u. The pair (S, u) is *strongly detectable* if it can be determined from input and output values of Σ whether the next stable state has been reached. □

To derive necessary and sufficient conditions for strong detectability, let f be the recursion function of Σ, and let s be its stable recursion function. Assume that Σ is at a stable combination with the state x when the input character changes to u, resulting in the chain of transitions $x = f^0(x,u), x^1 = f(x,u), x^2 = f^2(x,u), ..., x' = f^i(x,u)$, where $x' = s(x,u)$ is the next stable state (or set of states) of Σ. Letting $\setminus$ denote set difference, the set of transient states included in this chain of transitions is

$$f^-(x,u) := \left\{ \bigcup_{j=0,1,2,...,i} f^j(x,u) \right\} \setminus s(x,u).$$

Similarly, let $S \subset X$ be a set of states and let u be an input value that forms valid combinations with all states $x \in S$. Then, the collection of transient states included in transition chains triggered by pair (S, u) is

$$f^-[S,u] := \bigcup_{x \in S} f^-(x,u).$$

Now, the set $h[f^-[S,u]]$ includes all output values that are generated by transient states, while the set $h[s[S,u]]$ includes all output values that are generated by the stable states reached at the end of the transition process. Clearly, to be able to determine whether transitions have ceased, these two sets must be disjoint. A slight reflection shows that the converse is also true:

Proposition 2.4: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, let S be a set of states of Σ, and let $u \in A$ be an input character for which (x, u) is a valid combination for all $x \in S$. Then, (S, u) is strongly detectable if and only if $h[f^-[S,u]] \cap h[s[S,u]] = \varnothing$, the empty set. □

Example 2.5: Continuing with Example 2.1, consider $S = \{x^1, x^3\}$ and $u = a$. Then, $f^-[S,a] = \{x^2, x^3\}$ and $s[S,a] = \{x^1\}$, so that $h[s[S,a]] = \{0\}$ and $h[f^-[S,a]] = \{1\}$. Thus, $h[f^-[S,a]] \cap h[s[S,a]] = \varnothing$, and the pair $(\{x^1, x^3\}, a)$ is strongly detectable. A similar calculation shows that $(x^1, a), (x^1, b), (x^2, a), (x^2, b), (\{x^2, x^3\}, a), (\{x^2, x^3\}, b)$ are all strongly detectable. □

III. Generalized Realizations

Generalized realizations are an important tool for controlling asynchronous machines with critical races. First, some terminology. Two states x, x' of a machine $\Sigma = (A, Y, X, x_0, f, h)$ are *output equivalent* if they yield the same output values, i.e., if $h(x) = h(x')$. On a set S of states, we can induce an *output equivalence partition* $\{S_1, S_2, ..., S_p\}$ which consists of disjoint classes $S_1, S_2, ..., S_p$ of output equivalent states of S.

It is often convenient to group several states into one entity. For example, consider a critical race whose outcomes are all output equivalent. After such race, it is impossible to determine the exact state of the machine from input and output values. We group the set of all states that are consistent with available data into one entity that includes the uncertainty about the current state of the machine. This leads to the following notion. ($\#S$ denotes the cardinality of a set S and $P(S)$ is the family of all subsets of S.)

Definition 3.1: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, let χ be a set disjoint from X and including at least $2^{\#X}$ elements, and let $\Phi : P(X) \to X \cup \chi$ be an injective function satisfying $\Phi(x) = x$ for all states $x \in X$. With an output equivalent set $S \subset X$, associate the element $\xi := \Phi(S)$.

If $\#S > 1$, then ξ is a *group state* of Σ, while S is the *underlying set* of ξ and is denoted $S(\xi)$. For an input character $u \in A$, the pair (ξ, u) (or the pair (S, u)) is a *valid pair* if (x, u) is a valid pair for all $x \in S$.

An *extended state set* $\tilde{X}$ of Σ is the union of the original state set X with a set of group states. A *generalized state set* $\tilde{X}$ of Σ is an extended state set for which the following is true for all valid pairs $(\xi, u) \in \tilde{X} \times A$: every member of the output equivalence partition of the set $s[S(\xi), u]$ is either a single state or is represented by a group state in $\tilde{X}$. □

The stable recursion function s of $\Sigma = (A, Y, X, x_0, f, h)$ can be extended to group states. Let $\tilde{X}$ be a generalized state set of Σ. For a member $\zeta \in \tilde{X}$, denote by $S(\zeta)$ the underlying set of states. For a valid combination $(\zeta, u) \in \tilde{X} \times A$, let $\{S_1, ..., S_m\}$ be the output equivalence partition of the set $s[S(\zeta), u]$, and let $\zeta^i \in \tilde{X}$ be the generalized state associated with $S_i, i = 1, ..., m$. Then, the *generalized stable recursion function* $s_g : \tilde{X} \times A \to \tilde{X}$ and the *generalized output function* $h_g : \tilde{X} \to Y$ are defined by

$$s_g(\zeta, u) := \{\zeta^1, ..., \zeta^m\} \text{ and } h_g(\zeta) := h[S(\zeta)] \qquad (3)$$

for all $\zeta \in \tilde{X}$. Since $S(\zeta)$ is an output equivalence class, $h[S(\zeta)]$ is a single output character. Then, $\Sigma_g := (A, Y, \tilde{X}, x_0, s_g, h_g)$ is called a *generalized machine* associated with Σ. By construction, a generalized machine has exactly the same input/output behavior as the original machine Σ. Consequently, Σ_g is just another realization of Σ, and we refer to it as a *generalized realization.*

Considering that a generalized state represents a known output equivalence class, the generalized state of a machine is uniquely determined by the machine's input and output values, even after a critical race. This fact imparts the significance of generalized realizations: a generalized realization creates a deterministic relationship between input-output data and the generalized states of a possibly non-deterministic machine. The following algorithm, which follows from Definition 3.1, builds generalized realizations.

Algorithm 3.2: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, and assume that Σ has ρ critical race pairs $(r_1, v_1), (r_2, v_2), ..., (r_\rho, v_\rho)$. Let χ be a set that is disjoint from the state set X and has at least $2^{\#X}$ elements, and

let $\Phi : P(X) \to X \cup \chi$ be an injective function satisfying $\Phi(x) = x$ for all $x \in X$. The following steps build a generalized realization $\Sigma_g := (A, Y, \tilde{X}, x_0, s_g, h_g)$ of Σ.

Step 1. For every valid pair $(x, u) \in X \times A$ that is not a critical race, set $s_g(x, u) := s(x, u)$. If $\rho = 0$, then set $\Upsilon := \varnothing$ and go to Step 9.

Step 2. Define the ordered family of pairs $S := \{(r_1, v_1), (r_2, v_2), ..., (r_\rho, v_\rho)\}$ and the sets $\Upsilon := \varnothing$ and $\Upsilon' := \varnothing$. Assign $i := 1$ and $q := -1$.

Step 3. Let σ_i be the $i-th$ member of the family S, and let $\{G_1, ..., G_k\}$ be the output equivalence partition of the set of states $s(\sigma_i)$.

Step 4. Let $\xi_j := \Phi(G_j), j = 1, 2, ..., k$, and replace Υ by the set $\Upsilon \cup \{\xi_1, ..., \xi_k\}$. Assign $s_g(\sigma_i) := \{\xi_1, ..., \xi_k\}$, and denote $S(\xi_j) := G_j, j = 1, 2, ..., k$.

Step 5. If $i + 1 \leq \#S$, then replace i by $i + 1$ and return to Step 3.

Step 6. Define the difference set $\Upsilon'' := [\Upsilon \setminus \Upsilon'] \setminus X$; then replace $\Upsilon' := \Upsilon$.

Step 7. If $\Upsilon'' = \varnothing$, then go to Step 9.

Step 8. Replace S by an ordered family consisting of all valid pairs $(S(\zeta), u)$, where $\zeta \in \Upsilon''$ and $u \in A$, and return to Step 3.

Step 9. Terminate the Algorithm. The set Υ is the set of group states, $\tilde{X} := X \cup \Upsilon$ is the generalized state set, and s_g is the generalized stable recursion function of Σ. □

Generalized realizations are usually not minimal realizations. Nevertheless, they help in the process of designing output feedback controllers for asynchronous machines with critical races, since they create a deterministic relationship between the input/output features of a machine and its generalized state, even in the aftermath of a critical race. It can be verified that Algorithm 3.2 has polynomial computational complexity.

Example 3.3: We use Algorithm 3.2 to build a generalized realization of the machine Σ of Example 2.1. Here, there is one critical race $\sigma_1 = (x^1, b)$ with outcomes $s(\sigma_1) = s(x^1, b) = \{x^2, x^3\}$. As $h(x^2) = h(x^3) = 1$, the output equivalence partition of $s(\sigma_1)$ is the single class $G_1 = \{x^2, x^3\}$. Associating with G_1 the generalized state x^4, we set $s_g(\sigma_1) := x^4$. Further, $s_g(x^4, a) := s[G_1, a] = \{s(x^2, a), s(x^3, a)\} = \{x^1\}$; $s_g(x^4, b) := s[G_1, b] = \{s(x^2, b), s(x^3, b)\} = \{x^2, x^3\} = x^4$; and $h_g(x^4) := h(\{x^2, x^3\}) = 1$. At other valid combinations s_g is identical to s and h_g is identical to h; the transition table of Σ_g is

X	a	b	Y
x^1	x^1	x^4	0
x^2	x^1	x^2	1
x^3	x^1	x^3	1
x^4	x^1	x^4	1

□

We consider next the question of how to determine the current generalized state of an asynchronous machine.

IV. Observers

Following common control theoretic terminology, we use the term *observer* to refer to an asynchronous machine whose role is to estimate the state of another asynchronous machine. Specifically, we use observers to determine the most recent generalized state reached by an observed machine Σ. The observers considered here differ from the ones of GENG and HAMMER [2004 and 2005], since presently our observed machines are afflicted by critical races.

Consider a machine $\Sigma = (A, Y, X, x_0, f, h)$ associated with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$. An *observer* ϑ for Σ is a stable state input/state machine $\vartheta = (A \times Y, Z, z_0, \sigma)$ having two inputs: the input $u \in A$ of Σ and the output $y \in Y$ of Σ. The state set Z of ϑ consists of the same elements as the generalized state set $\tilde{X}$ of Σ, and the initial state of ϑ is identical to the initial state of Σ, i.e., $z_0 = x_0$. The recursion function $\sigma : Z \times A \times Y \to Z$ of ϑ is defined by

$$\sigma(z, u, y) := \begin{cases} \zeta \in s_g(z, u) \text{ if } y = h_g(\zeta) \text{ and} \\ \quad (z, u) \text{ is strongly detectable,} \\ z \text{ otherwise.} \end{cases} \tag{4}$$

The recursion function $\sigma(z, u, y)$ is well defined since, when starting from a strongly detectable pair (z, u), an output value $y \in h_g(s_g(z, u))$ is reached only when Σ arrives at its next stable generalized state (see Proposition 2.4). Furthermore, by construction, there is exactly one generalized state $\zeta \in s_g(z, u)$ that satisfies $y = h_g(\zeta)$ for the current output value y of Σ.

By this definition of the recursion function, the state of the observer ϑ switches to the state $\zeta \in s_g(z, u)$ immediately after the machine Σ has reached its next generalized state ζ. As the state of the observer ϑ is also its output, the output of ϑ tracks the most recent generalized state reached by Σ through a strongly detectable transition. Needless to say, Σ must be restricted to strongly detectable transitions.

To describe the operation of the observer ϑ in more detail, assume that Σ is at a generalized stable combination (x, v) when the input character changes to u, where (x, u) is a strongly detectable generalized pair. This change of the input character may give rise to a chain of transitions, ultimately leading Σ to the generalized state $\zeta \in s_g(x, u)$. As the pair (x, u) is strongly detectable, it follows by Proposition 2.4 that Σ displays the output value $y \in h_g(s_g(x, u))$ right upon reaching the state ζ, and not before. According to 4, the observer ϑ transitions to the state $\zeta \in Z$ when it detects the output value $y \in h_g(s_g(x, u))$, and it will linger at this state until the end of the next strongly detectable transition. As ϑ is an input/state machine, ζ becomes the new output value of ϑ. In this way, ϑ displays at its output the most recent generalized state reached by Σ through a strongly detectable transition.

V. Controllers

Recall that our ultimate objective is to control an asynchronous machine to eliminate the effects of critical races

and make the machine follow a prescribed deterministic model Σ'. Adopting the control strategy used by GENG and HAMMER [2004 and 2005] for output feedback control of race-free machines, we decompose the controller C of Figure 1 into two asynchronous machines: an observer ϑ and a control unit F, as follows.

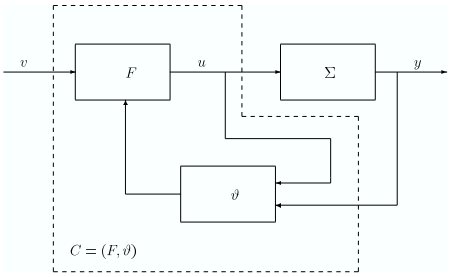


Fig. 2. Controller Decomposition

Here, the observer ϑ outputs the most recent generalized state visited by the machine Σ. This information is used by the control unit F to generate an input string that drives Σ to its next destination along the path necessary to match the response of the model Σ'. The controller C of Figure 1 is formed by the combination $C = (F, \vartheta)$.

The observer ϑ helps achieve fundamental mode operation of the closed loop control configuration of Figure 2. Indeed, as we have seen in section IV, the state and the output of ϑ stay constant while Σ is in transition - they only change upon Σ reaching its next generalized state. Thus, the control unit F receives constant input while Σ is in transition, and hence remains resting in its latest stable combination. Consequently, the entire controller $C = (F, \vartheta)$ remains in a stable combination while Σ is in transition (recall that the external input v is kept constant during transitions of the composite machine Σ_c).

The moment the machine Σ has reached its next stable combination via a strongly detectable transition, the observer ϑ undergoes a transition to its next stable state which is represented by the same symbol as the generalized stable state of Σ (see (4)). Upon reaching its next stable state, ϑ changes its output, inducing a change at the input of the control unit F. Depending on the transition function of F, this may or may not induce transitions in F. The important consequence of this chain of events is that transitions among the three machines ϑ, F, and Σ always occur sequentially - one machine at a time. According to Condition 2.2, this guarantees fundamental mode operation of composite machine of Figure 2.

VI. Reachability of Generalized Realizations

As we have just seen, fundamental mode operation of a machine-controller combination can be guaranteed only at strongly detectable pairs of the controlled machine. As a result, the operation of a controlled machine must be restricted to such pairs. This leads us to the following notion (see also PENG and HAMMER [2008]).

Definition 6.1: Let Σ be an asynchronous machine with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$, where $\tilde{X} = \{x^1, x^2, ..., x^\mu\}$. For a pair of generalized states $x^i, x^j \in \tilde{X}$, define the set of input characters

$$\alpha(x^i, x^j) : \quad = \begin{cases} a \in A : (x^i, a) \text{ is a strongly} \\ \quad \text{detectable pair and } x^j \in s_g(x^i, a). \end{cases} \tag{5}$$

Then, letting N be a character not in A, the *generalized one-step reachability matrix* $R_g(\Sigma)$ is a $\mu \times \mu$ matrix whose i, j entry is

$$R_{g_{ij}}(\Sigma) := \begin{cases} \alpha(x^i, x^j) \text{ if } \alpha(x^i, x^j) \neq \varnothing, \\ N \text{ if } \alpha(x^i, x^j) = \varnothing, \end{cases} \tag{6}$$

$i, j = 1, 2, ..., \mu$. □

Example 6.2: Considering the list of strongly detectable transitions of Example 2.5, the generalized one-step reachability matrix for Example 3.3 is

$$R_g(\Sigma) = \begin{pmatrix} a & N & N & b \\ a & b & N & N \\ a & N & b & N \\ a & N & N & b \end{pmatrix}. \square$$

In order to work with generalized one-step reachability matrices, we define a few specialized operations (see VENKATRAMAN and HAMMER [2006a, b, c]). Denoting by A^+ the set of all non-empty strings of characters of the alphabet A, consider two elements $w_1, w_2 \in A^+ \cup N$, where N is a character not in A. The operation $\Cup$ of *unison* is then defined by

$$w_1 \Cup w_2 := \begin{cases} w_1 \cup w_2 \text{ if } w_1, w_2 \in A^+; \\ w_1 \text{ if } w_1 \in A^+ \text{ and } w_2 = N; \\ w_2 \text{ if } w_1 = N \text{ and } w_2 \in A^+; \\ N \text{ if } w_1 = w_2 = N. \end{cases}$$

For two subsets $\sigma_1, \sigma_2 \subset A^+ \cup N$, the *unison* is defined by

$$\sigma_1 \Cup \sigma_2 := \{w_1 \Cup w_2 : w_1 \in \sigma_1 \text{ and } w_2 \in \sigma_2\}.$$

Given two $n \times n$ matrices A and B whose entries are subsets of $A^+ \cup N$, the *unison* $C := A \Cup B$ is the $n \times n$ matrix with the entries $C_{ij} := A_{ij} \Cup B_{ij}, i, j = 1, ..., n$. This operation is similar to numerical matrix addition, with N taking the role of the zero.

Next, an operation that mimics matrix multiplication: the *concatenation* of two elements $w_1, w_2 \in A^+ \cup N$ is

$$conc(w_1, w_2) := \begin{cases} w_2 w_1 \text{ if } w_1, w_2 \in A^+; \\ N \text{ if } w_1 = N \text{ or } w_2 = N. \end{cases}$$

For two subsets $W, V \subset A^+ \cup N$, the *concatenation* is

$$conc(W, V) := \Cup_{w \in W, v \in V} conc(w, v).$$

Then, for two $n \times n$ matrices C, D whose entries are subsets of $A^+ \cup N$, the *product* $Z := CD$ is an $n \times n$ matrix whose (i,j) entry Z_{ij} is

$$Z_{ij} := \biguplus_{k=1,2,...,n} conc(C_{ik}, D_{kj}), i, j = 1, ..., n.$$

With this product, we can define powers of the generalized one-step reachability matrix:

$$R_g^q(\Sigma) := R_g^{q-1}(\Sigma) R_g(\Sigma), q = 2, 3, ...$$

By construction, the (i,j) entry of $R_g^q(\Sigma)$ consists of all strings of q input characters that take Σ_g from a stable combination with x^i to a stable combination with x^j in exactly q stable and strongly detectable transitions.

To demonstrate the use of $R_g^q(\Sigma)$, let the string $u_1 u_2 ... u_q$ be a member of the (i,j) entry of $R_g^q(\Sigma)$. Then, we can proceed from x^i to x^j as follows: at a stable combination with the generalized state x^i, apply the input character u_1, and hold it until the observer ϑ of (4) displays a state of the set $s_g(x^i, u_1)$; then, apply the input character u_2, and again wait until the observer ϑ displays a state of the set $s_g(x^i, u_1 u_2)$; and so on for q steps. If this string of transitions includes critical races, then x^j is one of the possible outcomes.

Further, for an integer $q \in \{1, 2, ...\}$, define the matrix

$$R_g^{(q)}(\Sigma) := \biguplus_{r=1,...,q} R_g^r(\Sigma). \tag{7}$$

The (i,j) entry of $R_g^{(q)}(\Sigma)$, if not N, includes all input strings that may take Σ from a stable combination with the generalized state x^i to a stable combination with the generalized state x^j in q or fewer stable and strongly detectable steps. The following statement is similar to MURPHY, GENG, and HAMMER [2003, Lemma 3.9].

Lemma 6.3: Let Σ be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, x^2, ..., x^\mu\}$ and the generalized one-step reachability matrix $R_g(\Sigma)$. Then, the following are equivalent:

(i) The generalized state x^j is stably reachable from the generalized state x^i through a finite string of stable and strongly detectable transitions, possibly as one outcome of a critical race.

(ii) The (i,j) entry of the matrix $R_g^{(\mu-1)}(\Sigma)$ is not N. □

The Lemma shows that all strongly detectable stable transitions of the generalized realization of Σ are characterized by the *generalized stable reachability matrix* $\Gamma(\Sigma) := R_g^{(\mu-1)}(\Sigma)$.

Example 6.4: Using the matrix $R_g(\Sigma)$ of Example 6.2, a direct calculation yields

$$\Gamma(\Sigma) = \begin{pmatrix} \{a, ba, aba\} & N & N & \{ab, bab, b\} \\ \{a, ba, aba\} & b & N & \{ab, bab\} \\ \{a, ba, aba\} & N & b & \{ab, bab\} \\ \{a, ba, aba\} & N & N & \{ab, bab, b\} \end{pmatrix}. \square$$

As indicated earlier, the relationship between input values, output values, and generalized states of an asynchronous machine with critical races is similar to the relationship between inputs, outputs, and states of a deterministic asynchronous machine. Thus, once the generalized stable reachability matrix is available, we are in a situation similar to the one encountered in the deterministic case. We can then utilize the generalized stable reachability matrix in combination with the framework developed in GENG and HAMMER [2005] to derive output feedback controllers for asynchronous machines with critical races. Full details are provided in PENG and HAMMER [2008].

REFERENCES

[1] G. BARRETT and S. LAFORTUNE [1998] "Bisimulation, the supervisory control problem, and strong model matching for finite state machines," Discrete Event Dynamic Systems: Theory and Application, vol. 8, no. 4, pp. 377-429.

[2] M. D. DIBENEDETTO, A. SALDANHA and A. SANGIOVANNI-VINCENTELLI [1994] "Model matching for finite state machines," Proceedings of the IEEE Conference on Decision and Control, vol. 3, 1994, pp. 3117-3124.

[3] X. J. GENG and J. HAMMER [2004] "Asynchronous sequential machines: input/output control," Proceedings of the 12th Mediterranean Conference on Control and Automation, Kusadasi, Turkey, June 2004.

[4] X. J. GENG and J. HAMMER [2005] "Input/output control of asynchronous sequential machines," IEEE Transactions on Automatic Control, vol. 50, no. 12, pp. 1956-1970.

[5] J. HAMMER [1994] "On some control problems in molecular biology," Proceedings of the IEEE conference on Decision and Control, December 1994.

[6] J. HAMMER [1995] "On the modeling and control of biological signal chains," Proceedings of the IEEE conference on Decision and Control, December 1995.

[7] J. HAMMER [1996a] "On the corrective control of sequential machines," International Journal of Control, vol. 65, no. 2, pp. 249-276.

[8] J. HAMMER [1996b] "On the control of incompletely described sequential machines," International Journal of Control, vol. 63, no. 6, pp. 1005-1028.

[9] J. HAMMER [1997] "On the control of sequential machines with disturbances," International Journal of Control, vol. 67, no. 3, pp. 307-331.

[10] Z. KOHAVI [1970] "Switching and Finite Automata Theory," McGraw-Hill Book Company, New York.

[11] T. E. MURPHY, X. J. GENG and J. HAMMER [2002] "Controlling races in asynchronous sequential machines," Proceeding of the IFAC World Congress, Barcelona, July 2002.

[12] T. E. MURPHY, X. J. GENG and J. HAMMER [2003] "On the control of asynchronous machines with races," IEEE Transactions on Automatic Control, vol. 48, no. 6, pp. 1073-1081.

[13] J. PENG and J. HAMMER [2008] "Input/Output Control of Asynchronous Sequential Machines with Races", submitted for publication.

[14] P. J. G. RAMADGE and W. M. WONHAM [1987] "Supervisory control of a class of discrete event processes," SIAM Journal of Control and Optimization, vol. 25, no. 1, pp. 206-230.

[15] J. G. THISTLE and W. M. WONHAM [1994] "Control of infinite behavior of finite automata," SIAM Journal on Control and Optimization, vol. 32, no. 4, pp. 1075-1097.

[16] N. VENKATRAMAN and J. HAMMER [2006a] "Stable realizations of asynchronous sequential machines with infinite cycles," Proceedings of the 2006 Asian Control Conference, Bali, Indonesia, 2006.

[17] N. VENKATRAMAN and J. HAMMER [2006b] "Controllers for asynchronous machines with infinite cycles," Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, 2006.

[18] N. VENKATRAMAN and J. HAMMER [2006c] "On the control of asynchronous sequential machines with infinite cycles," International Journal of Control, vol. 79, no. 7, pp. 764-785.

[19] J.M. YANG and J. HAMMER [2007a] "State Feedback Control of Asynchronous Sequential Machines with Adversarial Inputs", International J. of Control (to appear).

[20] J.M. YANG and J. HAMMER [2007b] "Counteracting the Effects of Adversarial Inputs on Asynchronous Sequential Machines", Proceedings of the IFAC World Congress, Seoul, Korea, July 2008.