On the Control of Asynchronous Machines With Races

Thomas E. Murphy, Xiaojun Geng, and Jacob Hammer

***Abstract*—The problem of eliminating the effects of critical races on asynchronous machines is considered in a control theoretic context. State feedback controllers that eliminate the effects of critical races are developed. The results include necessary and sufficient conditions for the existence of such controllers and algorithms for their design. When the controllers exist, they eliminate the race effects and control the machine to match a given race-free model.**

***Index Terms*—Asynchronous machines, feedback, finite state machines, hazards, races.**

I. Introduction

A critical race is a flaw in the operation of an asynchronous sequential machine, causing the machine to exhibit unpredictable behavior. Critical races can occur as a result of malfunctions, design flaws, or implementation flaws. When a critical race occurs, common practice is to replace the machine with a new one, built from a race-free design. This note proposes a different, possibly more efficient, alternative: develop feedback controllers that control a race-afflicted machine to render its response predictable and acceptable. This eliminates the need to redesign, rebuild, and replace the entire machine. Instead, it is only necessary to add an appropriate feedback controller, an attractive option when replacement of the machine is not practical or not economical. The controller can be deployed before a flaw develops to yield a system that functions properly both before and after a malfunction occurs, improving reliability. The configuration is given in Fig. 1.

The controller C, sometimes called a *corrective controller*, drives the machine Σ to eliminate the effects of races and to guarantee a predictable and desirable response of the closed loop system. Our main objective is to find necessary and sufficient conditions for the existence of a corrective controller C. Whenever C exists, we provide an algorithm for its design. The input–output relation of the closed-loop system of Fig.1 is denoted by Σ_c.

Consider an asynchronous machine Σ afflicted by a critical race with q possible outcomes. To model the race, represent Σ by a *critical race family* $M = \{\Sigma^1, \ldots, \Sigma^q\}$ of q asynchronous machine models; Σ^i represents Σ for one outcome of the race (see the example in Section VI). The controller C controls Σ so that Σ_c has the same response, irrespective of which one of $\Sigma^1, \ldots, \Sigma^q$ is inserted for Σ. The closed-loop system response is then the same for all outcomes of the race, i.e., it is predictable.

Recall the distinction between *stable states* and *unstable states* of an asynchronous machine: a stable state is a state at which the machine lingers until an input change occurs; an unstable state is a state through which the machine passes rapidly, unable to rest there with the existing input. A machine spends (ideally) zero time at an unstable state, and, in its path from one stable state to another, it may pass in rapid succession through several unstable states. The latter are not noticeable to the user, as the machine does not linger at them. Our controller C transforms into unstable states all states at which the members of M differ from each other. The "noticeable" response of the closed-loop system is then the same for all members of M.

Manuscript received November 29, 2001; revised January 21, 2002. Recommended by Associate Editor L. Dai.

T. E. Murphy is with the School of Computing, Armstrong Atlantic State University, Savannah, GA 31419 USA.

X. Geng and J. Hammer are with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611 USA.

Digital Object Identifier 10.1109/TAC.2003.812814

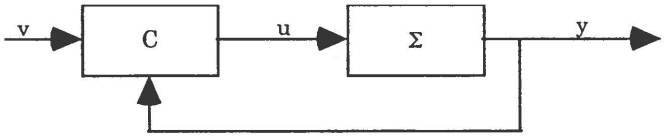

Fig. 1. Feedback configuration.

The existence of a corrective controller depends on certain reachability properties that all members of M have in common. These properties are characterized in terms of a numerical matrix called the "skeleton matrix" of M (Section III). Necessary and sufficient conditions for the existence of a corrective controller are stated in terms of the skeleton matrix.

This note continues [6]–[9], and is based on [17]. The existing literature on races concentrates on the design of race free machines [10]–[12], [16], [3], [2], [14]. There seems to be no previous literature on the use of control techniques to overcome the effects of critical races in an existing machine. Other aspects of discrete systems are examined in [1], [4], [6], [15], and [18]–[20], and others. These investigations do not address issues specifically related to asynchronous machines, like races, stable and unstable states, and fundamental mode operation (Section II).

The presentation here is for machines with a single critical race, but can be generalized to machines with multiple races. Basics are reviewed in Section II, and Section III introduces the skeleton matrix. Controllers are developed in Section IV and Section V; Section VI is an example.

II. Terminology and Background

A. Asynchronous Machines and States

An asynchronous machine is activated by input characters from a finite nonempty *alphabet* A. A *word* w over A is a finite (possibly empty) ordered string of characters of A; the *length* $|w|$ is the number of characters of w. The set of all words over A is A^*, while A^+ is the set of all nonempty words. The concatenation of two words w_1, $w_2 \in A^*$ is $w := w_1 w_2$. A *partial function* is a function defined over only a subset of its domain.

A machine Σ is determined by a quintuple (A, Y, X, f, h), where A is the *input alphabet*, Y is the *output alphabet*, and X is a finite set of *states*; f: $X \times A \rightarrow X$ and h: $X \times A \rightarrow Y$ are partial functions: f is the *recursion function* and h is the *output function*. A point $(x, u) \in X \times A$ at which f and h are defined is called a *valid pair*. An input sequence $u_0, u_1, u_2, \ldots \in A$ is transformed by Σ into an output sequence $y_0, y_1, y_2, \ldots, \in Y$ according to the relations $x_{k+1} = f(x_k, u_k)$, $y_k = h(x_k, u_k)$, $k = 0, 1, 2, \ldots$; the initial condition x_0 is specified. Here, k serves as a step counter, advancing when a change in state or input occurs. The input sequence is *permissible* if (x_k, u_k) is a valid pair for all k. A valid pair (x, u) is a *stable combination* if $f(x, u) = x$ (e.g., [13]). A state is *potentially stable* if it has a stable combination. By standard convention, all states of Σ are potentially stable. Our main interest is in *input/state machines*, namely, machines where $Y = X$ and

$$x_{k+1} = f(x_k, u_k),\ y_k = x_k, \qquad k = 0, 1, 2, \ldots. \tag{2.1}$$

B. Fundamental Mode Operation and Stable-State Machines

To prevent ambiguity of the response, it is best to restrict asynchronous machines to *fundamental mode operation*, where no more than one of their input and state variable may change at a time (e.g., [13]). Input variables change only one at a time, and only after the

machine has reached a stable combination (so the state stays fixed while an input value changes). In the following, all machines and all combinations of machines operate in fundamental mode.

When (x, u) is not a stable combination, the machine Σ continues from it through a chain of transitions that terminates if and only if a stable combination (x', u) with the same input value u occurs. If it exists, x' is the *next stable state* of x with the input value u. If there is no such x', then Σ has an infinite cycle. An infinite cycle cannot be terminated in fundamental mode operation, since u must be kept fixed while state transitions are in progress. Thus, for fundamental mode operation, one must exclude machines with infinite cycles; then, every valid pair (x, u) has a next stable state x' with input u.

Definition 2.2: Let $\Sigma = (A, Y, X, f, h)$ be a machine with no infinite cycles. The *stable recursion function* s of Σ is a partial function $s : X \times A \to X$ defined for every valid pair (x, u) of Σ by $s(x, u) := x'$, where x' is the next stable state of x with the input value u. The stable recursion function induces the *stable-state machine* (A, Y, X, s, h) of Σ, denoted $\Sigma_{|s}$. ♦

When going from one stable state to the next, Σ may pass through a number of unstable states that are ignored in practice, since Σ cannot linger at them. Thus, the operation of Σ is best described by its stable state machine $\Sigma_{|s}$. However, unstable states cannot be entirely ignored, since our controllers operate by transforming undesirable stable states into unstable states.

Consider an input string $w = u_0 u_1 \dots u_k \in A^+$ applied at a state x of Σ. In fundamental mode operation, the input value u_0 is kept fixed until Σ reaches the next stable state $s(x, u_0)$. Then, the input switches to u_1 and stays constant till the next stable state $(s(s(x, u_0), u_1))$ is reached; and so on to the last stable state $x' := s(\dots s(s(x, u_0), u_1), u_2) \dots, u_k) =: s(x, w)$. We review now the (standard) notion of machine equivalence (e.g., [5]).

Definition 2.3: A state x of $\Sigma = (A, Y, X, f, h)$ and a state x' of $\Sigma' = (A, Y, X', f', h')$ are *equivalent* if the following is true: When Σ starts from x and Σ' starts from x', then Σ and Σ' have the same permissible input strings, and, for each such string, they produce the same output string. The machines Σ and Σ' are *equivalent* (written $\Sigma = \Sigma'$) if every state of Σ has an equivalent state of Σ' and every state of Σ' has an equivalent state of Σ. ♦

C. Races and Race Families

A race occurs when two or more variables try to change values simultaneously (e.g., [13] and [21]). As simultaneous change of independent variables is unlikely, a sequential change in unpredictable order occurs instead. The order is unpredictable since it depends on random hardware conditions. In a *critical race*, the response of the machine depends on the order in which the race variables change, creating a pair (r, v) whose next stable state is unpredictable. We also call (r, v) a *critical race*. A critical race may derive from a component failure, an implementation flaw, or a design error.

Let Σ be a machine with a single critical race (r, v). When Σ is at the state r and receives the input v, the next state of the machine can be one of several options, say $\rho_1, \dots, \rho_q$, called the *outcomes* of the race. To represent Σ, build a family $M = \{\Sigma^1, \dots, \Sigma^q\}$ of q machines all having the same input set, the same output set, the same state set, and the same output function as Σ. The recursion function f_i of Σ^i is the same as the recursion function f of Σ, except at the critical race (r, v)

$$f_i(x, u) = \begin{cases} f(x, u), & \text{for all } (x, u) \neq (r, v) \\ \rho_i, & \text{for } (x, u) = (r, v), \text{ where } \rho_i \neq r \end{cases} \tag{2.4}$$

$i = 1, \dots, q$. We call M a *critical race family*; its members are deterministic. Each member represents Σ for one outcome of the race (see Section VI). If r_i is the next stable state of ρ_i with input v, then, $r_i \neq r$, $i = 1, \dots, q$; otherwise, an infinite cycle occurs.

Consider a machine Σ operating in fundamental mode, with recursion function f, initial state x_0, and input string $w = v_0 v_1 \dots v_{m-1}$. The machine goes through a string of states $x_0, \dots, x_m$, where $x_{i+1} = f(x_i, v_i)$, $i = 0, \dots, m-1$, ending with the stable combination (x_m, v_{m-1}). Fundamental mode operation implies that $v_i = v_{i+1}$ whenever (x_i, v_i) is not a stable combination, $i = 0, \dots, m-2$. This induces a list of pairs $P(\Sigma, x_0, w) := \{(x_0, v_0), (x_1, v_1), \dots, (x_{m-1}, v_{m-1}), (x_m, v_{m-1})\}$ called a *path*. A path is generated by the recursion function, not by the stable recursion function. The following is a consequence of (2.4).

Lemma 2.5: Let $M = \{\Sigma^1, \dots, \Sigma^q\}$ be a critical race family induced by a critical race (r, v), let x be a state, and let w be an input string. If $(r, v) \notin P(\Sigma^i, x, w)$ for an $i \in \{1, \dots, q\}$, then all paths are equal: $P(\Sigma^j, x, w) = P(\Sigma^i, x, w)$, for all $j = 1, \dots, q$. ♦

We will list only one representative of a group of "repeating" input values, so, e.g., $w = v_0 v_0 v_1 v_2 v_2 v_2$ will be listed $w = v_0 v_1 v_2$. This is, in fact, the string used by the hardware (and the stable state machine), where input is constant until change.

D. Controllers

The controller C of Fig.1 has two inputs: the reference v and the feedback y. Take the values of v in the input alphabet A of Σ. With Σ of (2.1), the input set of C is then $A \times X$. The output set of C must be in A, so $C = (A \times X, A, \Xi, \phi, \eta)$, where Ξ is the state set, ϕ is the recursion function, and η is the output function. The closed-loop system Σ_c, being composed of Σ and of C, has the state set $X \times \Xi$; let f_c be its recursion function and let h_c be its output function. As the output of Σ and Σ_c are the same, h_c: $(X \times \Xi) \times A \to X$: $h_c((x, \xi), v) = x$, for all $\xi \in \Xi$ and all valid pairs $(x, v) \in X \times A$ of Σ.

The fact that Σ of (2.1) is strictly causal guarantees that the closed-loop system of Fig.1 is well posed, i.e., all signals in the loop are uniquely and causally determined by v (e.g., [8]). To guarantee that no new critical races arise, the composite system of Fig.1 operates in fundamental mode. The latter occurs when C satisfies the following conditions.

i) Starting from a stable combination and in response to a change in the external input variable v of C, the output value of C does not change until C reaches its next stable state.
ii) In response to changes in the output of Σ, the controller C does not commence any state transitions until Σ reaches its next stable state.

Then, Σ and C do not engage simultaneously in state transitions; while one of Σ, C is in the process of state transitions, its input remains fixed; and not more than one input of C changes at a time. Note that, by standard guidelines, v must be kept constant sufficiently long between changes, to allow the composite system to reach its next stable combination.

For a member Σ^i of a critical race family $M = \{\Sigma^1, \dots, \Sigma^q\}$, let Σ_c^i be the closed loop system obtained when Σ is replaced by Σ^i in Fig. 1. Denote by $\Sigma_{c|s}^i$ the stable-state machine induced by Σ_c^i. Then, the main topic of our discussion is the following.

Model Matching Problem 2.6: Let $M = \{\Sigma^1, \dots, \Sigma^q\}$ be a critical race family and let Σ' be a race-free stable-state machine with the same input and output alphabets as the members of M. Find necessary and sufficient conditions for the existence of a controller C such that $\Sigma_{c|s}^i = \Sigma'$, for all $i = 1, \dots, q$. When C exists, provide a method for its design. ♦

The controller C of Problem 2.6 eliminates the effects of a race by assigning to the closed-loop system the desirable race-free behavior Σ' (see the example in Section VI). Model matching for synchronous machines was investigated in [6], [7], and [9]. Asynchronous machines,

however, allow more flexibility, since one only needs to match a stable state behavior (rather than a full behavior). Accordingly, the methods used here are different, as are the results.

III. Reachability

A. Stably Reachable States

In fundamental mode operation, a controller can act only when the machine it controls is in a stable combination. Consequently, we are interested in reachability properties of the stable-state machine.

Definition 3.1: Let x and x' be two potentially stable states of a machine Σ and let s be the stable recursion function of Σ. Then, x' is *stably reachable* from x if there is a permissible input string $u = u_0 u_1 \dots u_k$ of $\Sigma_{|s}$ for which $x' = s(x, u)$. ♦

Any stably reachable state can be reached in at most $(\#X - 1)$ steps, where $\#X$ is the number of states of the machine.

Lemma 3.2: Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable-state machine induced by an asynchronous machine Σ. If a state $x' \in X$ is stably reachable from the state $x \in X$, then there is an input string $u \in A^*$ of length $|u| \leq (\#X - 1)$ for which $x' = s(x, u)$.

Proof: Let x' be stably reachable from x and let $u' := u_0 u_1 \dots u_{k-1}$ be an input string for which $x' = s(x, u')$. If $|u'| \leq \#X - 1$, then the Lemma holds. If $k = |u'| > \#X - 1$, define the states $x_0, x_1, \dots, x_k$ by $x_0 := x$, $x_{i+1} := s(x_i, u_i)$, $i = 0, 1, \dots, k-1$, and $x_k = x'$. The list $x_0, \dots, x_k$ contains $k + 1 > \#X$ states, so at least one state must repeat in the list, say $x_i = x_j$, $0 \leq i < j \leq k$. Since $x_i = x_j$, the string $u'' := u_0 u_1 \dots u_{i-1} u_j \dots u_{k-1}$ (or $u'' := u_j \dots u_{k-1}$ when $i = 0$) still has $x' = s(x, u'')$, but $|u''| < |u'|$. This process of shortening the input string can be repeated until one obtains a string u of length $|u| < \#X$ still satisfying $x' = s(x, u)$. ♦

An input string u that takes a machine through the stable states $x_0, x_1, \dots, x_k$ induces a *cycle* if a state repeats in the list $x_0, x_1, \dots, x_k$. By the aforementioned proof, cycles can be eliminated by shortening u, without affecting the end states x_0 and x_k.

B. Matrix of Stable Transitions

Definition 3.3: Let Σ be a machine with input alphabet A, state set $X = \{x_1, \dots, x_n\}$, and stable recursion function s. The *one-step stable transition matrix* $R(\Sigma)$ of Σ is an $n \times n$ matrix whose (i, j) entry $R_{ij}(\Sigma)$ is the set of all (single) input characters $u \in A$ satisfying $x_j = s(x_i, u)$. If $R_{ij}(\Sigma)$ is empty, write $R_{ij}(\Sigma) := N$, where N is a character not in A. ♦

Clearly, $R(\Sigma)$ shows all one-step stable transitions of Σ; N indicates the lack of a corresponding one-step stable transition. To add N to our lexicon, define the set $A^*(N) := \{e | e \in A^* \text{ or } e = N\}$. Next, some special operations ($\cup$ marks union).

Definition 3.4: Let S_1 and S_2 be two nonempty subsets of $A^*(N)$. The *unison* $S_1 \Cup S_2$ is given by

$$S_1 \Cup S_2 := \begin{cases} (S_1 \cup S_2) \cap A^*, & \text{if } (S_1 \cup S_2) \cap A^* \neq \varnothing \\ N, & \text{otherwise.} \end{cases}$$

♦

The unison is similar to the union of sets, except that it removes the character N if the union contains a string of A^+.

Definition 3.5: For two elements of $w_1, w_2 \in A^*(N)$, the *concatenation* is

$$\operatorname{conc}(w_1, w_2) := \begin{cases} N, & \text{if } w_1 = N \text{ or } w_2 = N \\ w_1 w_2, & \text{otherwise.} \end{cases}$$

♦

Here, N takes the role of a "multiplicative zero." For subsets $W_1 = \{w_{1,1}, w_{1,2}, \dots, w_{1,q}\}, W_2 = \{w_{2,1}, w_{2,2}, \dots, w_{2,r}\} \subset A^*(N)$

$$\operatorname{conc}(W_1, W_2) := \Cup_{\substack{i=1,\dots,q \\ j=1,\dots,r}} \operatorname{conc}(w_{1,i}, w_{2,j}).$$

Definition 3.6: Let A and B be $n \times n$ matrices whose entries are subsets of $A^*(N)$. The *product* $C := AB$ is an $n \times n$ matrix whose i, j entry is $C_{ij} := \Cup_{k=1,\dots,n} \operatorname{conc}(A_{ik}, B_{kj}))$, $i, j = 1, \dots, n$. ♦

Note the similarity to numerical matrix multiplication. It can be verified that the (i, j) entry of a power $R^m(\Sigma)$ of the one-step stable transition matrix, if not N, consists of all input strings that take Σ from x_i to x_j in exactly m stable transitions.

Definition 3.7: Let A and B be $n \times n$ matrices with entries in $A^*(N)$. The *unison* $A \Cup B$ is an $n \times n$ matrix whose (i, j) entry is $(A \Cup B)_{ij} := A_{ij} \Cup B_{ij}$, $i, j = 1, \dots, n$. ♦

The operation of unison is analogous to the addition of numerical matrices. Using unison, define

$$R^{(m)}(\Sigma) := \Cup_{i=1,\dots,m} R^i(\Sigma), \qquad m = 2, 3, \dots. \tag{3.8}$$

The (i, j) entry of $R^{(m)}(\Sigma)$, if not N, consists of all input strings taking Σ from x_i to x_j in at most m stable transitions. So

$$R^*(\Sigma) := \Cup_{i \geq 1} R^i(\Sigma)$$

is an $n \times n$ matrix that characterizes all stable transition paths of Σ. Lemma 3.2 then leads to the following.

Lemma 3.9: The (i, j) entry of the matrix $R^*(\Sigma)$ is N if and only if the (i, j) entry of the matrix $R^{(n-1)}(\Sigma)$ is N. ♦

In other words, the matrix $R^{(n-1)}(\Sigma)$ characterizes all pairs of states (x_i, x_j) of Σ for which x_j is stably reachable from x_i.

Definition 3.10: The matrix $R^{(n-1)}(\Sigma)$ is called the *matrix of stable transitions* of Σ. ♦

The entries of $R^{(n-1)}(\Sigma)$, which characterize input strings for all stable transitions of Σ, are essential for the construction of controllers later. However, the existence of controllers can be determined from a simpler matrix introduced next.

C. Skeleton Matrix

Definition 3.11: Let $R(\Sigma)$ be the $n \times n$ one-step stable transition matrix of Σ. The *one-step skeleton matrix* $S(\Sigma)$ of Σ is an $n \times n$ matrix whose (i, j) entry is $S_{ij}(\Sigma) := 0$ if $R_{ij}(\Sigma) = N$ and $S_{ij}(\Sigma) := 1$ if $R_{ij}(\Sigma) \neq N$, $i, j = 1, \dots, n$. ♦

The one-step skeleton matrix is a matrix of zeros and ones. Its (i, j) entry is one if there is a one-step stable transition that takes Σ from the state x_i to the state x_j; otherwise, the (i, j) entry is zero. We need a special operation for skeleton matrices.

Definition 3.12: Let A, B be two $n \times n$ matrices of zeros and ones. The *combination* AB is again an $n \times n$ matrix of zeros and ones; its (i, j) entry is $(AB)_{ij} := \max\{A_{ik} B_{kj} : k = 1, \dots, n\}$, for all $i, j = 1, \dots, n$. ♦

Matrix combination is similar to matrix multiplication. Using combination, consider the mth "power" $S^m(\Sigma)$ of the one-step skeleton matrix $S(\Sigma)$, $m = 2, 3, \dots$. It can be seen that the (i, j) entry of $S^m(\Sigma)$ is 1 if and only if it is possible to reach x_j from x_i in exactly m stable transitions. For an integer $m \geq 1$, define the matrix $S^{(m)}(\Sigma)$ by setting its (i, j) entry to be

$$S_{ij}^{(m)}(\Sigma) := \max_{k=1,\dots,m} S_{ij}^k(\Sigma), \qquad m = 1, 2, \dots. \tag{3.13}$$

Then, $S^{(m)}(\Sigma)$ is also a matrix of zeros and ones, and $S^{(1)}(\Sigma) = S(\Sigma)$. A comparison of (3.13) and (3.8) shows that an entry of $S^{(m)}(\Sigma)$ is zero if and only if the corresponding entry of $R^{(m)}(\Sigma)$ is N. By

Lemma 3.9, this implies that a state x_j is stably reachable from a state x_i if and only if $S_{ij}^{(n-1)}(\Sigma) = 1$. Thus, $S^{(n-1)}(\Sigma)$ contains substantial structural information about Σ.

Definition 3.14: Let $S(\Sigma)$ be the $n \times n$ one-step skeleton matrix of Σ. The *skeleton matrix* of Σ is $K(\Sigma) := S^{(n-1)}(\Sigma)$. ♦

The following is a direct consequence of Lemma 3.9 and the correspondence between $S^{(m)}(\Sigma)$ and $R^{(m)}(\Sigma)$.

Proposition 3.15: For an asynchronous machine Σ with $n \geq 1$ states, $S^{(p)}(\Sigma) = S^{(n-1)}(\Sigma)$, for all $p \geq n-1$. ♦

For two $n \times n$ skeleton matrices K, K', write $K \geq K'$ whenever $K_{ij} \geq K'_{ij}$, for all $i, j = 1, \ldots, n$. The following is then true.

Proposition 3.16: Let $A \geq B$ and $C \geq D$ be $n \times n$ skeleton matrices. Then, under matrix combination, $AC \geq BD$.

Proof: Let $(AC)_{ij}$ and $(BD)_{ij}$ be the (i, j) entries of AC and BD, respectively. Using (3.12) and the relations $A \geq B$ and $C \geq D$, we obtain $(AC)_{ij} := \max\{A_{ik}C_{kj}: k = 1, \ldots, n\} \geq \max\{B_{ik}D_{kj}: k = 1, \ldots, n\} = (BD)_{ij}$, for all i, j. Hence, $AC \geq BD$. ♦

For critical race families, the next notion underlies our main results.

Definition 3.17: Let $M = \{\Sigma^1, \ldots, \Sigma^q\}$ be a critical race family with n states. The *skeleton matrix* $K(M)$ of M is an $n \times n$ matrix with the entries $K_{ij}(M) := \min\{K_{ij}(\Sigma^k): k = 1, \ldots, q\}$, $i, j = 1, \ldots, n$, where $K(\Sigma^k)$ is the skeleton matrix of Σ^k. ♦

Note that $K_{ij}(M) = 1$ if the state x_j is stably reachable from the state x_i for every member of M; otherwise, $K_{ij}(M) = 0$.

IV. Corrective Controllers: Basic Considerations

For Fig.1 with the input/state machine $\Sigma = (A, X, X, f, h)$ and the controller $C = (A \times X, A, \Xi, \phi, \eta)$, the closed-loop system is $\Sigma_c = (A, X, X \times \Xi, f_c, h_c)$. Let $\pi_x: X \times \Xi \times A \to X: (x, \xi, v) \mapsto x$ be the projection. Then, $h_c = \pi_x$, as the output of Σ_c is x. Given a stable state machine $\Sigma' = (A, X, X, s', h)$, the model matching problem for a race free machine seeks a controller C for which

$$\Sigma_{c|s} = \Sigma'. \tag{4.1}$$

Let $\gamma: X \times \Xi \times A \to X \times \Xi$ be the recursion function of $\Sigma_{c|s}$. By Definition 2.3, (4.1) amounts to the following. For every valid pair (x, v) of Σ', there is a state $\xi \in \Xi$ for which $((x, \xi), v)$ is a valid pair of $\Sigma_{c|s}$ and

$$\pi_x\gamma(x, \xi, v) = s'(x, v). \tag{4.2}$$

The next statement indicates how C can change the stable recursion function of Σ. Given two sets α, β, denote by $\beta\backslash\alpha$ the set of all elements of β that are not in α; denote by $s'[S]$ the image of a subset $S \subset X \times A$ through the function s'.

Theorem 4.3: Let $\Sigma_{|s} = (A, X, X, s, h)$ be the stable-state machine of the input/state machine Σ. Let $x_1 \times U_1, \ldots, x_k \times U_k \subset X \times A$, $k \geq 1$, be disjoint sets of valid pairs of Σ. Let $x'_i \in X$ be a state stably reachable by Σ from x_i, $i = 1, \ldots, k$. There is a controller C for which $\Sigma_{c|s}$ is equivalent to a stable-state machine $\Sigma' = (A, X, X, s', h)$ whose recursion function s' satisfies

i) $s'[x_i \times U_i] = x'_i$, for all $i = 1, \ldots, k$;

ii) $s'(z, t) = s(z, t)$, for all $(z, t) \in X \times A\backslash(\cup_{i=1,\ldots,k} x_i \times U_i)$.

Furthermore, the closed-loop system Σ_c is well posed and operates in fundamental mode.

Proof: Since x'_i is stably reachable from x_i, there is an input string $w_i := v_i^0 v_i^1 \ldots v_i^{m(i)-1} \in A^+$ for which $s(x_i, w_i) = x'_i$, $i = 1, \ldots, k$, where $m(i) := |w_i|$. With this input string, s generates the states

$$x_i^1 := s(x_i, v_i^0), \qquad x_i^{m(i)-1} := s\left(x_i^{m(i)-2}, v_i^{m(i)-2}\right)$$

$$x_i^2 := s(x_i^1, v_i^1), \ldots, \qquad x'_i := s\left(x_i^{m(i)-1}, v_i^{m(i)-1}\right)$$

$$i = 1, \ldots, k.$$

Let $U(x_i) \subset A$ be the set of all input characters that form stable combinations with the state x_i, $i = 1, \ldots, k$. Define the sets

$$S := \cup_{i=1,\ldots,k} x_i \times U(x_i). \qquad V := \cup_{i=1,\ldots,k} x_i \times U_i.$$

The controller $C = (A \times X, A, \Xi, \phi, \eta)$ can then be constructed as follows.

1) The state set Ξ of C has $2 + \Sigma_{i=1}^k m(i)$ states denoted by $\Xi = \{\xi_0, \xi_1, \xi_1^1, \xi_1^2, \ldots, \xi_1^{m(1)}, \xi_2^1, \ldots, \xi_2^{m(2)}, \ldots, \xi_k^1, \ldots, \xi_k^{m(k)}\}$.

2) The initial state of C is ξ_0, and C moves to the state ξ_1 upon detection of a stable combination with one of the states $x_1, \ldots, x_k$. To this end, the recursion function ϕ of C is defined at ξ_0 as follows:

$$\phi(\xi_0, (z, t)) := \xi_0, \text{ for all } (z, t) \in X \times A\backslash S$$

$$\phi(\xi_0, (x, u)) := \xi_1, \text{ for all } (x, u) \in S.$$

In fundamental mode, the output of C cannot change before C reaches the state ξ_1, so set $\eta(\xi_0, (z, t)) := t$, for all $(z, t) \in X \times A$, feeding into Σ the external input of Σ_c. At ξ_1, choose a character $u_i \in U(x_i)$ and set $\eta(\xi_1, (x_i, t)) := u_i$, for all $t \in A$, $i = 1, \ldots, k$. This preserves fundamental mode operation, since Σ was in a stable combination when the controller reached ξ_1.

3) Assume that an input value $u \in U_i$ appears at the input of Σ_c while Σ is in a stable combination with the state x_i. Then, C generates the input string w_i driving Σ to x'_i. In fundamental mode, w_i is generated one character at a time; a new character is generated after Σ reaches a stable combination. First, C signifies the detection of u by moving to the state ξ_i^1

$$\phi(\xi_1, (x_i, u)) := \xi_i^1, \text{ for all } u \in U_i, \qquad i = 1, \ldots, k$$

$$\phi(\xi_1, (x_i, u)) := \xi_1, \text{ for all } u \in U(x_i)\backslash U_i, \qquad i = 1, \ldots, k$$

$$\phi(\xi_1, (z, t)) := \xi_0, \text{ for all pairs } (z, t) \in X \times A\backslash(S \cup V).$$

Upon reaching the state ξ_i^1, the controller generates the first character of the input string w_i for Σ by setting $\eta(\xi_i^1, (z, t)) = v_i^0$, for all $(z, t) \in X \times A$, $i = 1, \ldots, k$. This makes Σ move to the state x_i^1.

4) Similarly, the following definitions prompt C to continue generating the input string w_i for Σ

$$\phi(\xi_i^j, (x_i^j, u)) := \xi_i^{j+1}, \text{ for all } u \in U_i$$

$$\phi(\xi_i^j, (z, t)) := \xi_i^j, \text{ for all } (z, t) \in X \times A\backslash x_i^j \times U_i$$

$$j = 1, 2, \ldots, m(i) - 1, \; i = 1, \ldots, k.$$

The state ξ_i^{j+1} of C signifies that Σ has reached the stable combination (x_i^j, v_i^{j-1}). In fundamental mode operation, Σ is now ready for the next input character v_i^j of w_i: for $j = 1, 2, \ldots, m(i) - 1$, set $\eta(\xi_i^{j+1}, (z, t)) := v_i^j$, for all $(z, t) \in X \times A$, $i = 1, \ldots, k$.

5) Finally, for $j = m(i)$, assign

$$\varphi\left(\xi_i^{m(i)}, (x_i'), u\right) := \xi_i^{m(i)} \text{ for } u \in U_i$$

$$\phi\left(\xi_i^{m(i)}, (z, t)\right) := \xi_0, \text{ for all } (z, t) \in X \times A \backslash S \quad i = 1, \ldots, k.$$

$$\phi\left(\xi_i^{m(i)}, (z, t)\right) := \xi_1, \text{ for } (z, t) \in S.$$

The definition of ϕ is consistent, since $x_1 \times U_1$, $x_2 \times U_2, \ldots, x_k \times U_k$ are disjoint sets. The recursion function γ of Σ_c is then

$$\gamma(x_i, \xi_0, u) = (s(x_i, u), \xi_1), \text{ for all } u \in U(x_i)$$

$$\gamma(x_i, \xi_1, u) = (x_i, \xi_1), \text{ for all } u \in U(x_i)$$

$$\gamma(z, \xi_0, t) = (s(z, t), \xi_0), \text{ for all } (z, t) \in X \times A \backslash S$$

$$\gamma(x_i, \xi_1, t) = (x_i, \xi_0), \text{ for all } t \in A \backslash (U(x_i) \cup U_i)$$

$$\gamma(x_i, \xi_1, u_i) = \left(x_i', \xi_i^{m(i)}\right), \text{ for all } u_i \in U_i$$

$$\gamma\left(x_i', \xi_i^{m(i)}, u\right) := \left(x_i', \xi_i^{m(i)}\right) \text{ for } u \in U_i$$

$$\gamma\left(z, \xi_i^{m(i)}, t\right) := (s(z, t), \xi_0), \quad \text{for all } (z, t) \in X \times A \backslash x_i' \times U_i$$

$i = 1, \ldots, k$, so that (4.2) is valid. By construction, the closed loop operates in fundamental mode. ♦

The controller C of Theorem 4.3 generates an input string that takes Σ from the state x_i to the state x_i' through a string of stable transitions of Σ. The intermediate states of this transition become unstable states of Σ_c, where x_i' becomes the next stable state of x_i with an input character of U_i. The number of states of C can be reduced by machine reduction techniques.

V. Model Matching

A. Model Matching for Deterministic Systems

Here is a solution of the model matching problem for deterministic asynchronous machines.

Theorem 5.1: Let $\Sigma = (A, X, X, f, h)$ be an input/state machine and let $\Sigma' = (A, X, X, s', h)$ be a stable-state input/state machine. The following two statements are equivalent.

i) There exists a controller C for which $\Sigma_{c|s} = \Sigma'$, where Σ_c is well posed and operates in fundamental mode.

ii) The skeleton matrices of the machines Σ and Σ' satisfy $K(\Sigma) \geq K(\Sigma')$.

Proof: Let γ be the stable recursion function of Σ_c. Define $s_c := \pi_x \gamma$, let Ξ be the state set of the controller C, and let $X = \{x_1, \ldots, x_n\}$ be the state set of Σ. Assume first that part i) of the Theorem holds. Then, for every valid pair $(x, u) \in X \times A$ of Σ', there is a state $\xi \in \Xi$ such that $s_c(x, \xi, u) = s'(x, u)$. Now, x and $s_c(x, \xi, u)$ are states of Σ, say $x = x_i \in X$ and $s_c(x, \xi, u) = s'(x, u) = x_k \in X$, so $x_k = s_c(x_i, \xi, u) = s'(x_i, u)$. For the one-step skeleton matrix $S(\Sigma')$ of Σ', this implies that $S_{ik}(\Sigma') = 1$.

Since C accesses only the input of Σ, the equality $x_k = s_c(x_i, \xi, u)$ means that there is an input string $w \in A^+$ such that $x_k = s(x_i, w)$, so that $K_{ik}(\Sigma) = 1$. Thus, $K_{ik}(\Sigma) = 1$ if $S_{ik}(\Sigma') = 1$, and $K(\Sigma) \geq S(\Sigma')$. Multiplying each side of this inequality by itself $n-1$ times, we obtain by Proposition 3.16 that $K^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma') = K(\Sigma')$. Applying Proposition 3.15, we conclude that $K(\Sigma) \geq K(\Sigma')$. This shows that i) implies ii). Conversely, assume that part ii) of the Theorem is valid; since always $K(\Sigma') \geq S(\Sigma')$, we obtain $K(\Sigma) \geq S(\Sigma')$. This means that, for every valid pair (x, u) of Σ', the state $s'(x, u)$ is stably reachable from x by Σ. Theorem 4.3 shows then the existence of a controller C for which $\Sigma_{c|s} = \Sigma'$, and part ii) implies part i). ♦

An algorithm for the construction of a controller C that satisfies Theorem 5.1 is provided in the proof of Theorem 4.3.

B. Controlling Races

Here is our solution of the model matching problem 2.6 for a system with a critical race.

Theorem 5.2: Let $M = \{\Sigma^1, \ldots, \Sigma^q\}$ be a critical race family of input/state machines and let Σ' be a stable-state input/state machine having the same state set and input alphabet as the members of M. Let $K(M)$ be the skeleton matrix of the family M, and let $K(\Sigma')$ be the skeleton matrix of Σ'. Then, the following two statements are equivalent.

1) There is a controller C for which $\Sigma_{c|s}^i = \Sigma'$, for all $i = 1, \ldots, q$, where the closed-loop systems $\Sigma_c^1, \Sigma_c^2, \ldots, \Sigma_c^q$ are all well posed and operate in fundamental mode.

2) The skeleton matrices satisfy $K(M) \geq K(\Sigma')$.

The proof of Theorem 5.2 (provided later) includes a construction of C; an example is in Section VI. In crude terms, C operates as follows. When C detects the stable combination (r_i, v) after the race, it applies an input string that drives the member Σ^i to a stable combination common to all members of M. This equalizes the stable state response of Σ_c, for all members of M.

The skeleton matrix $K(M)$ of a critical race family M cannot be entirely zeros, since each state is potentially stable. Thus, there always is a stable-state machine Σ' with skeleton matrix equal to $K(M)$. By Theorem 5.2, this entails that every critical race can be resolved via state feedback control. We turn now to technical issues related to the proof of Theorem 5.2.

Let s^i be the stable recursion function of a member Σ^i of M and let s' be the stable recursion function of Σ'. The set

$$D(\Sigma^i, \Sigma') = \{(x, u) \in X \times A: (x, u) \text{ is a valid pair of } \Sigma' \text{ and } s^i(x, u) \neq s'(x, u)\} \quad (5.3)$$

is the *discrepancy set* of Σ^i; it consists of all valid pairs for which the next stable state of Σ^i differs from that of Σ'. Here, controller action is necessary to match the desired response. When $K(\Sigma^i) \geq K(\Sigma')$ there is, for each $(x, u) \in D(\Sigma^i, \Sigma')$, a string $w \in A^+$ satisfying $s^i(x, w) = s'(x, u)$. Let $S_i(x, u) \subset A^+$ be the set of all such strings w. For a pair $(x, u) \in D(\Sigma^i, \Sigma')$ and an input string $w \in S_i(x, u)$, let $P(\Sigma^i, x, w)$ be the path of Σ^i. Recalling that (r, v) is the critical race of M, build the subset

$$D_N(\Sigma^i, \Sigma') := \{(x, u) \in D(\Sigma^i, \Sigma'): (r, v) \notin P(\Sigma^i, x, w) \text{ for at least one string } w \in S_i(x, u)\} \subset D(\Sigma^i, \Sigma') \quad i = 1, \ldots, q.$$

On $D_N(\Sigma^i, \Sigma')$, the stable state response of Σ^i can match the response of Σ' without passing (r, v). But then, by Lemma 2.5, every member of M can match the response of Σ' on $D_N(\Sigma^i, \Sigma')$ without passing through the race (r, v). This yields the following.

Lemma 5.4: In the notation of Theorem 5.2, assume that $K(M) \geq K(\Sigma')$. Then, $D_N(\Sigma^i, \Sigma') = D_N(\Sigma^j, \Sigma')$, for all $j = 1, \ldots, q$. ♦

In view of Lemma 5.4, we use the notation

$$D_N(M, \Sigma') := D_N(\Sigma^i, \Sigma'), i = 1, \ldots, q. \tag{5.5}$$

To match Σ' over $D_N(M, \Sigma')$, the controller can apply the same input string to all members of M. On the difference set

$$D_R(\Sigma^i, \Sigma') := D(\Sigma^i, \Sigma') \backslash D_N(M, \Sigma') \tag{5.6}$$

the response of Σ' cannot be matched by Σ^i without passing through the critical race (r, v). Still, by Lemma 2.5, the path of Σ' up to (but not beyond) (r, v) is identical for all members of M. This leads to the following.

Lemma 5.7: In the notation of Theorem 5.2, assume that $K(M) \geq K(\Sigma')$. Then, for every $(x, u) \in D_R(\Sigma^i, \Sigma')$, there is an input string w that takes all members of M from (x, u) to the critical race (r, v), meeting (r, v) only in the last step of w. ♦

Thus, the process of matching Σ' over $D_R(\Sigma^i, \Sigma')$ splits into two parts: i) before the race and ii) after the race. Before the race, one does not need to identify which member of M is active, since all members can use the same input. After the race, the string that the controller needs to generate may vary from one member of M to another. By reading the outcome of the race, the controller can identify which member of M is active and then select the appropriate input string. This action makes the next stable combination of Σ' into the postrace stable combination of all members of M; it is the basis of the next proof.

Proof (of Theorem 5.2): If $K(M) \not\geq K(\Sigma')$, then there is a machine $\Sigma^i \in M$ for which $K(\Sigma^i) \not\geq K(\Sigma')$. Then, by Theorem 5.1, there can be no controller C for which $\Sigma^i_{c|s} = \Sigma'$. Whence, part 1) of Theorem 5.2 implies part 2). Conversely, assume that part 2) of Theorem 5.2 is valid. We construct below a controller C satisfying part i) of Theorem 5.2, using earlier notation.

If $D(\Sigma^i, \Sigma') = \emptyset$, for all $i = 1, \ldots, q$, then all members of M match Σ', so there is no need for C. Otherwise, $D(\Sigma^i, \Sigma') \neq \emptyset$ for at least one i. Let $\pi_x D(\Sigma^i, \Sigma')$ be the set of all states x for which there is a u satisfying $(x, u) \in D(\Sigma^i, \Sigma')$. Let $U_i(x)$ be the set of all input characters forming stable combinations with the state x of Σ^i. The only pair at which the members of M differ from each other is (r, v), which, by (2.4), cannot be a stable combination. Hence, $U_i(x)$ is the same for all $i = 1, \ldots, q$; set $U(x) := U_i(x)$. Then, $V_i := \{x \times U(x)\colon x \in \pi_x D(\Sigma^i, \Sigma')\}$ consists of all stable combinations with states in $\pi_x D(\Sigma^i, \Sigma')$. Define

$$\begin{aligned} V &:= \cup_{i=1,\ldots,q} V_i \\ D_R(M, \Sigma') &:= \cup_{i=1,\ldots,q} D_R(\Sigma^i, \Sigma') \\ D &:= \cup_{i=1,\ldots,q} D(\Sigma^i, \Sigma') \\ D(x) &:= \{u \in A\colon (x, u) \in D\}. \end{aligned} \tag{5.8}$$

Assume that $D_N(M, \Sigma') \neq \emptyset$. For each $(x, u) \in D_N(M, \Sigma')$ there is, by definition, a (shortest) input string $w(x, u)$ satisfying $s(x, w(x, u)) = s'(x, u)$ and $(r, v) \notin P(\Sigma, x, w(x, u))$, where s is the common restriction of $s^1, \ldots, s^q$. Let $m(x, u) := |w(x, u)|$ and let $v^0(x, u), v^1(x, u), \ldots, v^{m(x,u)-1}(x, u)$ be the characters of $w(x, u)$, i.e.,

$$w(x, u) := v^0(x, u) \ldots v^{m(x,u)-1}(x, u). \tag{5.9}$$

The stable states through which $w(x, u)$ takes the machine Σ are denoted by

$$x^1(x, u) := s(x, v^0(x, u))$$

$$x^2(x, u) := s(x^1(x, u), v^1(x, u)), \ldots$$

$$x^{m(x,u)-1}(x, u) := s(x^{m(x,u)-2}(x, u), v^{m(x,u)-2}(x, u))$$

and

$$s'(x, u) := s(x^{m(x,u)-1}(x, u), v^{m(x,u)-1}(x, u)). \tag{5.10}$$

Define $n(N) := \#D_N(M, \Sigma')$ and, when $n(N) > 0$, let $(x_1, u_1), \ldots, (x_{n(N)}, u_{n(N)})$ be the elements of $D_N(M, \Sigma')$. The following set will be used later as part of the state set of the controller C (if $D_N(M, \Sigma') = \emptyset$, set $\Xi_N := \emptyset$):

$$\begin{aligned} \Xi_N := \Big\{ &(\xi_1(x_1, u_1), \ldots, \xi_{m(x_1,u_1)}(x_1, u_1), \ldots, \\ &\xi_1\left(x_{n(N)}, u_{n(N)}\right), \ldots, \xi_{m(x_{n(N)}, u_{n(N)})} \\ &\cdot \left(x_{n(N)}, u_{n(N)}\right) \Big\}. \end{aligned} \tag{5.11}$$

Next, set $n(R) := \#D_R(M, \Sigma')$. When $n(R) > 0$, consider an i with $D_R(\Sigma^i, \Sigma') \neq \emptyset$. For each $(x, u) \in D_R(\Sigma^i, \Sigma')$, there is then a (shortest) input string $w(x, u)$ such that $s^i(x, w(x, u)) = s'(x, u)$; the path $P(\Sigma^i, x, w(x, u))$ contains the critical race (r, v) of Σ. Eliminating cycles, we obtain that $P(\Sigma^i, x, w(x, u))$ contains (r, v) exactly once. Divide the path $P(\Sigma^i, x, w(x, u))$ into two parts $P_{1,i}(x, u)$ and $P_{2,i}(x, u)$: $P_{1,i}(x, u)$ contains (r, v) and ends with the stable combination (r_i, v) immediately following (r, v); the part $P_{2,i}(x, u)$ consists of the remaining segment of $P(\Sigma^i, x, w(x, u))$, starting with (r_i, v). Referring to Lemma 5.7, let

$$w_0(x, u) := v^0(x, u) \ldots v^{m(x,u)-1}(x, u) \tag{5.12}$$

be a shortest common input string taking Σ from x to (r, v). Here, $m(x, u) := |w_0(x, u)|$, and note that v is the last character of $w_0(x, u)$. The path $P(\Sigma^i, x, w_0(x, u))$ of Σ^i starts at (x, u), is driven by $w_0(x, u)$, and ends with (r_i, v). By Lemma 5.7, $P_{1,i}(x, u)$ can be replaced by $P(\Sigma^i, x, w_0(x, u))$; the concatenation $P(\Sigma^i, x, w_0(x, u))P_{2,i}(x, u)$ still takes Σ^i from (x, u) to the desired state $s'(x, u)$. Let

$$w_i(x, u) := v^0(x, u, i), \ldots, v^{m(x,u,i)-1}(x, u, i) \tag{5.13}$$

be the input string generating $P_{2,i}(x, u)$, where $m(x, u, i) := |w_i(x, u)|$. Since Σ^i is at r_i when $w_i(x, u)$ starts, we have $r_i = s^i(x^{m(x,u)-1}(x, u), v^{m(x,u)-1}(x, u))$, and $w_i(x, u)$ drives Σ^i through the states

$$\begin{aligned} x^1(x, u, i) &:= s^i(r_i, v^0(x, u, i)) \\ x^2(x, u, i) &:= s^i(x^1(x, u, i), v^1(x, u, i)), \ldots \\ s'(x, u) &:= s^i\left(x^{m(x,u,i)-1}(x, u, i), v^{m(x,u,i)-1}(x, u, i)\right), \end{aligned}$$

$$\text{for all } (x, u) \in D_R(\Sigma^i, \Sigma') \text{ and all } i = 1, \ldots, q. \tag{5.14}$$

The following will be used as states of the controller C (denote $D_R(M, \Sigma') = \{(x_1, u_1), \ldots, (x_{n(R)}, u_{n(R)})\}$ when not empty):

$$\Xi_R := \Big\{(\xi_1(x_1, u_1), \ldots, \xi_{m(x_1,u_1)}(x_1, u_1), \ldots, \xi_1(x_{n(R)}, u_{n(R)}), \ldots, \xi_{m(x_{n(R)},u_{n(R)})} \cdot (x_{n(R)}, u_{n(R)})\Big\} \tag{5.15}$$

$$\Xi_i := \Big\{(\xi_{1,i}(x_1, u_1), \ldots, \xi_{m(x_1,u_1,i),i}(x_1, u_1), \ldots, \xi_{1,i}(x_{n(R)}, u_{n(R)}), \ldots, \xi_{m(x_{n(R)},u_{n(R)},i),i} \cdot i(x_{n(R)}, u_{n(R)})\Big\}, \quad i = 1, \ldots, q. \tag{5.16}$$

Selections are so that $\Xi_N, \Xi_R, \Xi_1, \ldots, \Xi_q$ are disjoint sets. If $D_R(\Sigma^i, \Sigma') = \emptyset$, set $\Xi_i := \emptyset$. If $D_R(M, \Sigma') = \emptyset$, set $\Xi_R := \emptyset$ and $\Xi_i := \emptyset$, for all $i = 1, \ldots, q$. Finally, the set $\Xi_0 := \{\xi_0, \xi_1\}$ of two more elements will also be used later for states of C.

We are ready to start the construction of the controller C. First, the state set of C is given by

$$\Xi := \Xi_0 \cup \Xi_N \cup \Xi_R \cup \Xi_1 \cup \cdots \cup \Xi_q \tag{5.17}$$

and X is the state set of Σ. In the construction that follows, the states of Ξ_0 help C record certain stable combinations of Σ, to ensure fundamental mode operation. The states of Ξ_N and Ξ_R help C generate input strings of Σ along path segments common to all members of M. Finally, the states of Ξ_i help the controller generate the input string of Σ after the machine has passed through the race with the outcome r_i. The recursion function ϕ and the output function η of C are constructed in the following steps, assuming that $\Xi_N, \Xi_R, \Xi_1, \ldots, \Xi_q$ are all nonempty; adjustments can be made for other cases.

i) Initially, C is at the state ξ_0. Without changing its output, C moves to the state ξ_1 when it detects a pair in V, so Σ is in a stable combination before its input is changed by C (fundamental mode operation). Accordingly, ϕ is defined by

$$\phi(\xi_0, (z, t)) := \xi_0, \text{ for all } (z, t) \in X \times A \backslash V$$

$$\phi(\xi_0, (x, u)) := \xi_1, \text{ for all } (x, u) \in V.$$

The controller is inactive in the state ξ_0, applying to Σ its own external input. Accordingly, the output function η is defined by $\eta(\xi_0, (z, t)) := t$, for all $(z, t) \in X \times A$. At the state ξ_1, choose a character $u^* \in U(x)$, and set $\eta(\xi_1, (z, t)) := u^*$, for all $(z, t) \in X \times A$. Then, Σ remains in a stable combination at x as long as C is in the state ξ_1 (fundamental mode operation).

ii) Assume that Σ is in a stable combination at a state $x \in \pi_x D$, and an input value u appears for which $(x, u) \in D$. Then, C prepares to generate an input string taking Σ to the state $s'(x, u)$, to match Σ'. This process is similar to the one used in the proof of Theorem 4.3. First, C moves to a state $\xi_1(x, u)$ to signify the encounter of (x, u) (fundamental mode operation)

$$\phi(\xi_1, (x, u)) := \xi_1(x, u) \text{ for } (x, u) \in D$$

$$\phi(\xi_1, (x, u)) := \xi_1, \text{ for all } u \in U(x) \backslash D(x)$$

and

$$\phi(\xi_1, (z, t)) := \xi_0, \text{ for all pairs } (z, t) \in X \times A \backslash (V \cup D).$$

Upon reaching $\xi_1(x, u)$, the controller starts to generate the string $w(x, u)$ or $w_0(x, u)$, as needed. This is accomplished by setting $\eta(\xi_1(x, u), (z, t)) = v^0(x, u)$, for all $(z, t) \in X \times A$ [by (5.9) or (5.12)]. Note that Σ is in a stable combination when this change in its input occurs. The input $v^0(x, u)$ moves Σ to the state $x^1(x, u)$. The pair $(x^1(x, u), v^0(x, u))$ is again a stable combination, being the next step of s. Then, C continues to generate $w(x, u)$ [or $w_0(x, u)$] as input for Σ, according to iii).

iii)

$$\phi(\xi_j(x, u), (x^j(x, u), u)) := \xi_{j+1}(x, u)$$

$$\phi(\xi_j(x, u), (z, t)) := \xi_j(x, u), \text{ for all } (z, t) \neq (x^j(x, u), u)$$

$$\eta(\xi_{j+1}(x, u), (z, t)) := v^j(x, u), \text{ for all } (z, t) \in X \times A.$$

$$1 \leq j \leq m(x, u) - 1, \text{ and } (x, u) \in D$$

where $v^j(x, u)$ is from (5.9) or (5.12), as the case may be.

iv) For $j = m(x, u)$, consider two cases: a) When $(x, u) \in D_N(M, \Sigma')$, assign

$$\phi(\xi_{m(x,u)}(x, u), (s'(x, u), u)) := \xi_{m(x,u)}(x, u)$$

$$\phi(\xi_{m(x,u)}(x, u), (z, t)) := \xi_0, \text{ for all } (z, t) \neq (s'(x, u), u)$$

completing the action of C here, since $s'(x, u)$ has been reached. b) When $(x, u) \in D_R(\Sigma^i, \Sigma')$ for some $i \in \{1, \ldots q\}$, set

$$\phi(\xi_{m(x,u)}(x, u), (r_i, u)) := \xi_{1,i}(x, u)$$

$$\phi(\xi_{m(x,u)}(x, u), (z, t)) := \xi_{m(x,u)}(x, u), \text{ for all } (z, t) \neq (r_i, u)$$

$$\eta(\xi_{1,i}(x, u), (z, t)) := v^0(x, u, i), \text{ for all } (z, t) \in X \times A$$

here, $v^0(x, u, i)$ is the first character of (5.13). Then, C continues to generate (5.13), following the stable combinations (5.14):

$$\phi(\xi_{j,i}(x, u), (x^j(x, u, i), u)) := \xi_{j+1,i}(x, u)$$

$$\phi(\xi_{j,i}(x, u), (z, t)) := \xi_{j,i}(x, u), \text{ for all } (z, t) \neq (x^j(x, u, i), u))$$

$$\eta(\xi_{j+1,i}(x, u), (z, t)) := v^j(x, u, i), \text{ for all } (z, t) \in X \times A,$$

$$j = 1, \ldots, m(x, u, i) - 1.$$

The end of the input string (5.13) is reached at $j = m(x, u, i) - 1$; by (5.14), Σ^i reaches then the desired state $s'(x, u)$, so assign

$$\phi(\xi_{m(x,u,i),i}(x, u), (s'(x, u), u)) := \xi_{m(x,u,i),i}(x, u)$$

$$\phi(\xi_{m(x,u,i),i}(x, u), (z, t)) := \xi_0, \text{ for all } (z, t) \neq (s'(x, u), u).$$

This completes the construction of C, yielding a closed-loop system that is well posed and operates in fundamental mode. ♦

The construction of the controller C in the proof does not attempt to minimize the number of states. Once C has been derived, its number of states can be reduced by state reduction techniques (e.g., [13]). Careful

selection of the strings generated by the controller can further reduce the number of controller states. If necessary, unstable transitions can be eliminated from the output of the closed loop system by adding an output function similar to the one used for C in the proof.

VI. Example

We demonstrate the construction of the controller C of Theorem 5.2. The machine Σ to be controlled has the input alphabet $A = \{a, b, c\}$ and the state set $X = \{x_0, x_1, x_2\}$. There is a critical race at (x_0, c) with the outcomes x_1 and x_2

Σ: f

	a	b	c
x_0	x_0	x_1	$\{x_1, x_2\}$
x_1	x_2	x_1	x_1
x_2	x_2	x_1	x_2

.

Σ induces a critical race family $M = \{\Sigma^1, \Sigma^2\}$ of two members whose stable recursion functions s^1 and s^2 are

$\Sigma^1_{|s}$: s^1

	a	b	c
x_0	x_0	x_1	x_1
x_1	x_2	x_1	x_1
x_2	x_2	x_1	x_2

$\Sigma^2_{|s}$: s^2

	a	b	c
x_0	x_0	x_1	x_2
x_1	x_2	x_1	x_1
x_2	x_2	x_1	x_2

. (6.1)

From the tables, the one-step stable transition matrices $R(\Sigma^1)$, $R(\Sigma^2)$, and the one step skeleton matrices $S(\Sigma^1)$, $S(\Sigma^2)$ are

$$R(\Sigma^1) = \begin{pmatrix} \{a\} & \{b, c\} & N \\ N & \{b, c\} & \{a\} \\ N & \{b\} & \{a, c\} \end{pmatrix}$$

$$R(\Sigma^2) = \begin{pmatrix} \{a\} & \{b\} & \{c\} \\ N & \{b, c\} & \{a\} \\ N & \{b\} & \{a, c\} \end{pmatrix}$$

$$S(\Sigma^1) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad S(\Sigma^2) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The skeleton matrices $K(\Sigma^1)$ and $K(\Sigma^2)$, calculated from Definition 3.14, yield the skeleton matrix $K(M)$ the family M as follows:

$$K(\Sigma^1) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad K(\Sigma^2) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$K(M) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

Choose the stable-state machine $\Sigma' := \Sigma^2_{|s}$ as the model to match (this is an arbitrary choice), and follow the proof of Theorem 5.2 to obtain a controller C. By (5.3) and (6.1), $D(\Sigma^1, \Sigma') = \{(x_0, c)\}$ and, since $\Sigma' = \Sigma^2$, we get $D(\Sigma^2, \Sigma') = \emptyset$. The set of input values that form stable combinations with the state x_0 is $U(x_0) = \{a\}$, and $V = \{(x_0, a)\}$ [see (5.8)]. From (6.1), the recursion function s' of $\Sigma' = \Sigma^2_{|s}$ has $s'(x_0, c) = x_2$, and the set of input strings that take Σ^1 from (x_0, c) to $s'(x_0, c) = x_2$ is $S_1(x_0, c) = \{ca\}$; the set of input strings that take Σ^2 from (x_0, c) to $s'(x_0, c)$ is $S_2(x_0, c) = \{c\}$. The paths are

$$P(\Sigma^1, x_0, ca) = \{(x_0, c), (x_1, c), (x_1, a), (x_2, a)\}$$

$$P(\Sigma^2, x_0, c) = \{(x_0, c), (x_2, c)\}.$$

Both paths include the race (x_0, c), so $D_N(M, \Sigma') = \emptyset$, $D_R(\Sigma^1, \Sigma') = D(\Sigma^1, \Sigma') = \{(x_0, c)\}$. Using (5.11), (5.15) and (5.16), we obtain $\Xi_N = \emptyset$, $\Xi_R = \{\xi_1(x_0, c)\}$, $\Xi_1 = \{\xi_{1,1}(x_0, c)\}$, and $\Xi_2 = \emptyset$. By (5.17), C has the state set $\Xi = \{\xi_0, \xi_1, \xi_1(x_0, c), \xi_{1,1}(x_0, c)\}$, i.e., four states. Following the proof of Theorem 5.2, the recursion function ϕ and the output function η of C are

$$\phi(\xi_0, (z, u)) := \xi_0, \text{ for all } (z, u) \in X \times A \backslash (x_0, a)$$

$$\phi(\xi_1(x_0, c), (x_1, c)) := \xi_{1,1}(x_0, c)$$

$$\phi(\xi_0, (x_0, a)) := \xi_1$$

$$\phi(\xi_1(x_0, c), (z, t)) := \xi_1(x_0, c), \text{ for all } (z, t) \notin \{(x_1, c), (x_2, c)\}$$

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

$$\phi(\xi_1(x_0, c), (x_2, c)) := \xi_0$$

$$\phi(\xi_1, (x_0, c)) := \xi_1(x_0, c)$$

$$\eta(\xi_1(x_0, c), (z, u)) := c, \text{ for all } (z, u) \in X \times A$$

$$\phi(\xi_1, (x_0, a)) := \xi_1$$

$$\phi(\xi_{1,1}(x_0, c), (x_2, c)) := \xi_{1,1}(x_0, c)$$

$$\phi(\xi_1, (z, t)) := \xi_0, \text{ for all pairs } (z, t) \notin \{(x_0, a), (x_0, c)\}$$

$$\phi(\xi_{1,1}(x_0, c), (z, t)) := \xi_0, \text{ for all } (z, t) \neq (x_2, c)$$

$$\eta(\xi_1, (z, u)) := a, \text{ for all } (z, u) \in X \times A$$

$$\eta(\xi_{1,1}(x_0, c), (z, u)) := a, \text{ for all } (z, u) \in X \times A.$$

References

[1] G. Barrett and S. Lafortune, "Bisimulation, the supervisory control problem, and strong model matching for finite state machines," *J. Discrete Event Dyna. Syst.*, vol. 8, no. 4, pp. 377–429, 1998.

[2] T. Chu, "Synthesis of hazard-free control circuits from asynchronous finite state machines specifications," *J. VLSI Signal Processing*, vol. 7, no. 1/2, pp. 61–84, 1994.
[3] P. K. Datta, S. K. Bandyopadhyay, and A. K. Choudhury, "A graph theoretic approach for state assignment of asynchronous sequential machines," *Int. J. Electron.*, vol. 65, no. 6, pp. 1067–1075, 1988.
[4] M. D. Dibenedetto, A. Saldanha, and A. Sangiovanni-Vincentelli, "Model matching for finite state machines," in *Proc. IEEE Conf. Decision Control*, 1994, pp. 3117–3124.
[5] S. Eilenberg, *Automata, Languages, and Machines*. New York: Academic, 1974.
[6] J. Hammer, "On some control problems in molecular biology," in *Proc. IEEE Conf. Decision Control*, Dec. 1994, pp. 4098–4103.
[7] ——, "On the modeling and control of biological signaling chains," in *Proc. IEEE Conf. Decision Control*, Dec. 1995, pp. 3742–3752.
[8] ——, "On corrective control of sequential machines," *Int. J. Control*, vol. 65, pp. 249–276, 1996.
[9] ——, "On the control of incompletely described sequential machines," *Int. J. Control*, vol. 63, pp. 1005–1028, 1996.
[10] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 161–190, 1954.
[11] ——, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, pp. 275–303, 1954.
[12] ——, "The design and use of hazard-free switching networks," *J. Assoc. Comput. Mach.*, vol. 4, no. 1, pp. 47–62, 1957.
[13] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1970.
[14] L. Lavagno, C. W. Moon, and A. Sangiovanni-Vincentelli, "Efficient heuristic procedure for solving the state assignment problem for event-based specifications," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 45–60, Jan. 1995.
[15] F. Lin, "Robust and adaptive supervisory control of discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 38, pp. 1848–1852, Dec. 1993.
[16] G. Maki and J. Tracey, "A state assignment procedure for asynchronous sequential circuits," *IEEE Trans. Comput.*, vol. C-20, pp. 666–668, June 1971.
[17] T. E. Murphy, "On the control of asynchronous sequential machines with races," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Florida, Gainesville, FL, 1996.
[18] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis, "Stability and stabilizability of discrete event systems," *J. ACM*, vol. 38, pp. 730–752, 1991.
[19] P. J. G. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.
[20] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," *SIAM J. Control Optim.*, vol. 32, no. 4, pp. 1075–1097, 1994.
[21] S. H. Unger, "Hazards, critical races, and metastability," *IEEE Trans. Comput.*, vol. 44, pp. 754–768, June 1995.