European Journal of Control (2012)3: 286-300
© 2012 EUCA
DOI:10.3166/EJC.18.286-300

European Journal of Control

Bursts and Output Feedback Control of Non-Deterministic Asynchronous Sequential Machines

Jun Peng, Jacob Hammer*

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA

The use of bursts—fast outbursts produced by an asynchronous sequential machine during transition—can enhance the capabilities of output feedback controllers to overcome uncertainties when controlling non-deterministic asynchronous sequential machines. The utilization of bursts involves the development of a new class of generalized realizations. The latter form the basis for the derivation of output feedback controllers that achieve deterministic model matching for non-deterministic machines. Necessary and sufficient conditions for the existence of such controllers are presented in terms of a numerical matrix derived from available data. Design outlines are also included.

Keywords: Asynchronous sequential machines, nonlinear feedback control, realization theory.

1. Introduction

Asynchronous sequential machines have received considerable attention in the scientific literature for more than half a century, as they form the building blocks of high-speed digital computing systems, are used to model parallel computing environments, and represent processes in molecular biology ([5]). Component failures, implementation flaws, design flaws, or genetic flaws in biology may cause asynchronous machines to develop critical races and become non-deterministic (e.g., [17]).

Until recent years, no general methodology was available for correcting the faulty operation of asynchronous sequential machines with critical races. The prevailing approach was to discard faulty machines and replace them by redesigned and newly built ones. This approach is often economically inefficient and may be infeasible when the defective machine is inaccessible or irreplaceable, as would be the case, for example, for a senescent biological process. An attractive alternative is to correct the functionality of a defective machine by adding a feedback controller.

In Fig. 1, the asynchronous machine Σ is controlled by an output feedback controller C, another asynchronous machine. The resulting closed loop machine is denoted by Σ_c. Our objective is to achieve *model matching*, namely, to find a controller C for which Σ_c emulates a specified model Σ'. As Σ' is deterministic, such a controller C eliminates the impact of the uncertainties caused by critical races present in Σ.

In general, an asynchronous machine has two kinds of states: *stable states*—states at which the machine rests until a change occurs in its input; and *transient states*—states the machine traverses quickly (ideally, in zero time). Often, during a transition from one stable state to another, an asynchronous machine passes through several transient states. For a user, the operation of an asynchronous machine is determined by stable states, as the machine does not linger at transient states.

During a transition from one stable state to another, the machine Σ may generate an output burst—a quick

*Correspondence to: J. Hammer, E-mail: hammer@mst.ufl.edu

Received 14 September 2010; Accepted 8 April 2011
Recommended by S. Martinez, A.J. van der Schaft

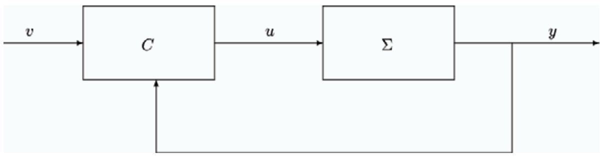

Fig. 1. The control configuration.

succession of output characters representing the transient states passed along the way. Although a burst occurs (ideally) in zero time, a properly equipped controller can take advantage of the burst by recording it in a sequential memory register and using it to gain information about the status of Σ. This allows the controller to achieve broader design objectives.

Output feedback control of asynchronous sequential machines with critical races was examined in [13, 14] under the requirement that no bursts be used: controllers were restricted to using only the current output value of the controlled machine Σ. At the cost of a small increase in controller complexity, the present paper develops controllers with enhanced capabilities by incorporating the use of bursts. In section 2, we demonstrate by an example that the controllers developed in this paper can accomplish control tasks that are impossible for the controllers of [13, 14]. Thus, the use of bursts does enhance controller capabilities.

When an asynchronous machine is in transition, it may pass quickly and asynchronously through a succession of transient states. If an input change occurs during such a transition, it will occur at an unpredictable transient state of the machine. As the response of a machine generally depends on its state, an input change at an unpredictable state may result in an unpredictable response. Thus, to assure consistent behavior of an asynchronous machine, it is best to keep the input constant while the machine is in transition. When this rule is observed, we say that the machine operates in *fundamental mode* (e.g., [10]). For the control configuration of Fig. 1, fundamental mode operation requires one of the machines Σ or C to rest while the other machine is in transition. In formal terms, we have

Condition 1.1: The configuration of Fig. 1 operates in fundamental mode when the following are valid:

(i) The machine Σ is in a stable state while the controller C is in transition;
(ii) The controller C is in a stable state while the machine Σ is in transition;
(iii) The external input v is constant while Σ or C are in transition. □

The present discussion is written within the general framework of [11, 12, 3, 4, 18, 19, 20 13, 14, 21]. These studies examine issues related to the control of asynchronous machines, such as the use of state feedback to eliminate the effects of critical races; the use of output feedback to control deterministic asynchronous machines; the use of state feedback to eliminate the effects of infinite cycles; the use of output feedback to eliminate the effects of critical races; and the use of state feedback to eliminate the effects of adversarial inputs. The present discussion investigates the utilization of bursts in output feedback control of non-deterministic asynchronous machines. We demonstrate that the use of bursts endows controllers with broader capabilities to overcome the impacts of uncertainties caused by critical races (see, for instance, Example 2.5 below).

Additional studies dealing with the control of sequential machines can be found in [15] and [16], where the theory of discrete event systems is explored; in [5–9, 2, 1], where model matching for discrete systems is explored; in the references cited in these studies, as well as elsewhere. The studies listed in this paragraph do not take into consideration specialized issues related to the operation of asynchronous sequential machines, such as the distinction between stable and transient states and fundamental mode operation.

The present paper is organized as follows. Section 2 formally introduces bursts and their treatment while section 3 develops a generalized theory of realization that facilitates a simple derivation of necessary and sufficient conditions for model matching. Section 4 deals with the construction of observers that utilize bursts. Output feedback controllers that use bursts to achieve model matching for asynchronous machines with critical races are developed in section 5. Finally, concluding remarks are provided in section 6.

2. Bursts and Detectability

2.1. Preliminaries

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, $f : X \times A \to X$ is a partial function called the *recursion function*, and $h : X \to Y$ is the *output function*. We denote by A^* the set of all strings of characters of A and by A^+ the set of all such strings that are not empty. A *strict prefix* of a string $a_1a_2 \ldots a_m \in A^+$ is a string of the form $a_1a_2 \ldots a_q$ with $1 \le q < m$.

The machine Σ accepts input sequences $u_0u_1 \ldots \in A^+$ and, in response, generates state sequences $x_0x_1 \ldots \in X^+$ and output sequences $y_0y_1 \ldots \in Y^+$ according to the recursion

$$\Sigma : \quad \begin{aligned} x_{k+1} &= f(x_k, u_k), \\ y_k &= h(x_k), k = 0, 1, 2, \ldots \end{aligned} \tag{2.1}$$

The *step counter k* advances by one upon a change of the input or of the state. The machine Σ is an *input/state machine* if $Y = X$ and $y_k = x_k$ for all $k = 0, 1, 2, \dots$ Here, we use the Moore representation of an asynchronous machine, so the output function h is independent of the input u_k (e.g., [10]).

A pair $(x, u) \in X \times A$ is a *valid pair* of Σ if the recursion function f is defined at it. A valid pair (x, u) that satisfies $x = f(x, u)$ is called a *stable combination*. The machine Σ lingers at a stable combination until the input character is changed. If (x, u) is not a stable combination, then Σ generates a chain of transitions $x_1 = f(x, u), x_2 = f(x_1, u), \dots$ which may or may not terminate. If the chain terminates, then there is a state $x_i \in X$ such that

$$\begin{aligned} &x_1 = f(x, u), x_2 = f(x_1, u), \dots, \\ &x_i = f(x_{i-1}, u), x_i = f(x_i, u), \end{aligned} \tag{2.2}$$

and (x_i, u) forms a stable combination of Σ. The state x_i is then the *next stable state* of x with the input u. If the chain (2.2) does not terminate, then it forms an *infinite cycle*. In the present paper, we consider only machines with no infinite cycles, so there is always a next stable state.

As indicated earlier, transient states of an asynchronous machine are not noticeable by users, since a machine passes through transient states very quickly (ideally, in zero time). Thus, from a user's perspective, the behavior of an asynchronous machine is determined by its stable states. To characterize the behavior associated with stable states, we define the *stable recursion function s* of Σ by $s(x, u) := x'$, where x' is the next stable state of (x, u). The *stable state machine* associated with Σ is $\Sigma_{|s} := (A, Y, X, x_0, s, h)$.

Two asynchronous machines Σ and Σ' are *stably equivalent* if the stable state machines associated with them are equal, i.e., if $\Sigma_{|s} = \Sigma'_{|s}$. As discussed earlier, stably equivalent machines are indistinguishable to their user. Consequently, we write $\Sigma = \Sigma'$ when Σ and Σ' are stably equivalent. We can now phrase the objective of our discussion in formal terms.

Problem 2.1: Model Matching: Given two asynchronous machines Σ and Σ', with Σ' serving as a model, find necessary and sufficient conditions for the existence of a controller C for which $\Sigma_c = \Sigma'$. If such a controller C exists, derive a method for its design. □

Next, a *critical race* is a pair $(r, v) \in X \times A$ for which the next stable state can be one of, say, $m > 1$ states $r^1, r^2, \dots, r^m$ called the *outcomes* of the critical race. At a critical race, the stable recursion function s of Σ is set-valued

$$s(r, v) := \{r^1, r^2, \dots, r^m\}.$$

When a transition chain involves critical races, then the symbols $x_1, x_2, \dots$ in (2.2) may represent sets of states. A set of states $S \subset X$ and an input character $u \in A$ form a *valid pair* (S, u) if (x, u) is a valid pair for all $x \in S$.

We turn now to the notion of bursts ([4]). In an asynchronous environment, it is impossible to distinguish between consecutive identical characters of a string, since the arrival of a repeated character is not a distinguishable event. Thus, a string of characters such as *aabbbcc* is indistinguishable from the string *abc*, since there is no specified time duration to signify the end of one instance and the beginning of the next instance of the same character. Repeated instances of the same character in a string are indistinguishable from a single appearance of that character. This leads us to the following notion.

Definition 2.2: Let $y_1, \dots, y_q \in Y$ be a set of characters satisfying $y_{i+1} \neq y_i$ for all $i = 1, \dots, q-1$. Then, the *burst* of a string $y = y_1 y_1 \dots y_1 y_2 y_2 \dots y_2 \dots y_q y_q \dots y_q$ is $\beta(y) := y_1 y_2 \dots y_{q-1} y_q$, i.e., the string obtained by removing all repeats of consecutive characters.

For a valid pair (x, u) of an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$, let $x_1 \in f(x, u), x_2 \in f(x_1, u), \dots, x_m \in f(x_{m-1}, u), x_m = f(x_m, u)$ be a string of transitions generated by Σ from (x, u) and ending at the stable state x_m. Then, the corresponding *burst* is the string of output characters $\beta(x, u, x_1 x_2 \dots x_m) := \beta\big(h(x)h(x_1)h(x_2) \dots h(x_{m-1})h(x_m)\big)$. □

Consider now the transition chain $x, x_1, x_2, \dots, x_m$ of Definition 2.2, and set $x' := x_m$. Due to critical races that may occur along the way, it might be possible to reach from the state x to the state x' via a different chain of transitions, say one passing through the states $x, x'_1, \dots x'_q$, where $x'_q = x'$. This chain of transitions may result in a different burst $\beta(x, u, x'_1 x'_2 \dots x'_q)$ generated during the transition from x to x'. Thus, in general, a transition from one state to another may be associated with multiple bursts, depending on the outcomes of critical races encountered along the way. We denote all these bursts by the symbol $\beta(x, u, x')$, namely,

$$\beta(x, u, x') := \begin{cases} \beta(x, u, x'_1 x'_2 \dots x'_q) & \text{if } x'_1 x'_2 \dots x'_q \text{ are the states of a transition chain from } x \text{ to } x' \text{ with the input character } u; \\ \varnothing & \text{if there is no transition from } (x, u) \text{ to } x'. \end{cases}$$

Here, $\varnothing$ is the empty set. A critical race pair (r, v) of Σ with the outcomes $r^1, r^2, \ldots, r^m$ induces the set of bursts

$$\beta(r, v) := \bigcup_{i=1,\ldots,m} \beta(r, v, r^i).$$

The following notation is also needed. For a transition chain of $q \geq 1$ steps through the states $x_1, x_2, \ldots x_q$, the burst induced by the first $q - 1$ steps is denoted by

$$\beta_{-1}\big(h(x_1)h(x_2)\ldots h(x_q)\big) := \begin{cases} \beta\big(h(x_1)h(x_2)\ldots h(x_{q-1})\big) \text{ for } q > 1, \\ \varnothing \text{ for } q = 1. \end{cases}$$

2.2. Detectability

To achieve fundamental mode operation of the configuration of Fig. 1, we must make sure that Condition 1.1 is satisfied. In particular, the controller C cannot change its output character before the machine Σ has reached its next stable state. Thus, C must determine from input and output data of the machine Σ whether or not Σ has reached a stable state. This determination, which can be aided by information extracted from bursts of Σ, must be made despite the possible presence of critical races in Σ. The need to determine whether Σ has reached a stable state leads us to the following notion ([3, 4, 13, 14]).

Definition 2.3: Let S be a set of states of an asynchronous machine Σ. Assume that Σ is at a stable combination with an unspecified member $x \in S$, when the input character of Σ switches to u, where (S, u) is a valid pair. The machine Σ is *detectable* at (S, u) if it is possible to determine from u and the output burst of Σ whether Σ has reached its next stable state. □

To discuss conditions for detectability, consider first a deterministic transition of the machine Σ through the states $x_1, x_2, \ldots, x_q$, from a stable combination with x_1 to a stable combination with x_q. In order for this transition to be detectable, we must be able to determine when its burst terminates. As discussed in [3, 4], such a determination is possible if and only if

$$\beta_{-1}\big(h(x_1)h(x_2)\ldots h(x_q)\big) \neq \beta\big(h(x_1)h(x_2)\ldots h(x_q)\big), \tag{2.3}$$

namely, if and only if there is a character change at the end of the burst. However, when uncertainty is present, the situation is somewhat more involved. Indeed, consider the following example.

Assume that the machine Σ rests at one of two states, say x' or x'', without it being known whether Σ is at x' or at x'' (this would imply that x' and x'' produce the same output character). Let u be an input character that forms valid pairs with both x' and x''; let x'_1 be the next stable state of (x', u) and let x''_1 be the next stable state of (x'', u). Further, let $\beta' := y_1y_2y_3$ be the burst generated by (x', u), and let $\beta'' := y_1y_2y_3y_4$ be the burst generated by (x'', u). Then, we have $\beta'_{-1} = y_1y_2$ and $\beta''_{-1} = y_1y_2y_3$, so that $\beta'_{-1} \neq \beta'$ and $\beta''_{-1} \neq \beta''$, and (2.3) holds for both transitions. However, when Σ presents the burst $y_1y_2y_3$, it is impossible to tell whether Σ has reached the stable state x'_1 or whether Σ is on its way to the stable state x''_1, as it is not known whether Σ started from x' or from x''. Thus, (2.3) is not a sufficient condition for detectability when uncertainty is present. This example leads us to necessary and sufficient conditions for detectability, as follows. First, some notation. Given a subset of states $S \subset X$ and an input character u for which (S, u) is a valid pair, denote by

$$B(S, u) := \bigcup_{x \in S, x' \in s(x,u)} \beta(x, u, x') \tag{2.4}$$

the set of all bursts that can be generated from states of S by the input character u.

Proposition 2.4: *Let S be a set of states of an asynchronous machine Σ, let $u \in A$ be an input character that forms a valid pair with S, and let $B(S, u)$ be the set of all bursts that can be generated by (S, u). Then, (S, u) is detectable if and only if the following are true for every burst $b \in B(S, u)$:*

(i) $b_{-1} \neq b$, and
(ii) b is not a strict prefix of any burst in $B(S, u)$.

Proof: Let Σ be at a stable combination with an unspecified state $x \in S$ when the input character switches to u, and let $x, x_1, \ldots, x_m$ be the resulting string of state transitions. Then, the burst is $b := \beta\big(h(x)h(x_1)\ldots h(x_m)\big) \in B(S, u)$. Assume first that conditions *(i)* and *(ii)* are valid. Then, by *(i)*, we can determine the end of the burst b (see [4, Proposition 23]), and by *(ii)* there is no burst in $B(S, u)$ that continues beyond the end of b. Hence, Σ has ceased transitions at the end of the burst b. In other words, Σ has reached its next stable state at the end of b, and, as this is the case for every member $b \in B(S, u)$, we conclude that the pair (S, u) is detectable. Thus, *(i)* and *(ii)* imply detectability.

Conversely, assume that (S, u) is detectable. Then, it follows directly by [4, Proposition 23] that *(i)* must be valid. Now, assume, by contradiction, that *(ii)* is not valid. Then, $B(S, u)$ includes two bursts of the form $b_1 := y_1y_2\ldots y_q$ and $b_2 := y_1y_2\ldots y_t$, with $t > q$. In such case, at the step q, it is not possible to tell whether Σ has reached the end of the burst b_1 (i.e., has reached a stable combination) or is just progressing within the burst b_2 (i.e., has not reached a stable combination yet). Hence, it cannot be determined from the burst whether Σ has reached its next stable state, in contradiction to our assumption that (S, u) is detectable. Thus, *(ii)* must be valid, and our proof concludes. □

The following example demonstrates testing for detectability. Note that the case presented in the example is not strictly detectable in the sense of [13], where only current output values (rather than output bursts) of the controlled machine are employed. Thus, the use of bursts does enhance control capabilities.

Example 2.5: Consider an asynchronous machine Σ with the input alphabet $A = \{a, b\}$, the output alphabet $Y = \{0, 1\}$, the state set $X = \{x^1, x^2, x^3, x^4\}$, and the transition table:

Table 1. The transition table of Σ.

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

For the set $S := \{x^2, x^3\}$, we can see from the table that the pair (S, b) is valid. Further, the pair (S, b) induces the transitions $x^2 \to x^4 \to x^1$ and $x^3 \to x^4 \to x^1$, with the output bursts $\beta(x^2, b, x^1) = 010$ and $\beta(x^3, b, x^1) = 010$. Thus, $B(S, b) = \{010\}$, and it follows by Proposition 2.4 that (S, b) is detectable. In contrast, it can be verified that (S, b) is not strongly detectable in the sense of [13]. Thus, the use of bursts enhances control capabilities. □

3. Generalized Realizations

In the present section, we broaden the notion of generalized realization ([20, 13, 14]) to accommodate the use of bursts. We start by introducing an equivalence relation on a machine's state set, under which states are considered equivalent if the process of reaching them generates the same burst.

Definition 3.1: Let Σ be an asynchronous machine with the stable recursion function s, let $S \subset X$ be a set of states, and let $u \in A$ be an input character for which (S, u) is a valid pair. For a burst β, the set of *burst equivalent states* $S(\beta)$ induced by S is the set of all states $x' \in s(S, u)$ for which $\beta \in \beta(x, u, x')$ for some state $x \in S$. □

When a transition of Σ starts from an unspecified state within the set S and produces a burst β on its way to the next stable state, the latter will be a member of the set $S(\beta)$ of burst equivalent states. It is not possible to determine from input and output data at which member of the set $S(\beta)$ the machine rests after the transition.

Example 3.2: For the set S of Example 2.5, we have $S(010) = x^1$. Consequently, there is no uncertainty after the transition in this case. □

We adapt now to our present setup the notion of generalized state ([20, 13, 14]). Below, $\#X$ denotes the cardinality of a set X, and $P(X)$ is the power set of X (the family of all subsets of X).

Definition 3.3: 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 with 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 a burst equivalent set of states S, associate the element $\xi := \Phi(S)$.

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

An *extended state set* 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 burst equivalent subset of $s(S(\xi), u)$ is either a single state or is represented by a group state in $\tilde{X}$. □

With a generalized realization, there is associated a recursion function defined as follows.

Definition 3.4: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, and 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, where $S(\zeta) := \zeta$ when $\zeta \in X$. For a valid pair $(\zeta, u) \in \tilde{X} \times A$, let $\{S_1, \ldots, S_m\}$ be the family of all burst equivalent subsets of states in $s[S(\zeta), u]$, and let $\zeta^i \in \tilde{X}$ be the generalized state associated with $S_i, i = 1, \ldots, m$. Then, the *generalized stable recursion function* s_g is defined by

$$s_g(\zeta, u) := \{\zeta^1, \ldots, \zeta^m\} \text{ for all valid pairs } (\zeta, u) \in \tilde{X} \times A. \tag{3.1}$$

The *generalized output function* $h_g : \tilde{X} \to Y$ is

$$h_g(\zeta) := h[S(\zeta)] \text{ for all } \zeta \in \tilde{X}. \tag{3.2}$$

The sextuple $\Sigma_g := (A, Y, \tilde{X}, x_0, s_g, h_g)$ forms a *generalized realization* of Σ. □

Note that, since $S(\zeta)$ in (3.2) is a burst equivalent set of states, all states of $S(\zeta)$ produce the same output value (equal to the last burst character); as a result, $h[S(\zeta)]$ is a single character. An important consequence is that the generalized realization Σ_g has the same input/output behavior as the original machine Σ. In other words, Σ_g is simply another realization of Σ.

The generalized state of a machine is uniquely determined by the burst of the most recent stable state transition, since each generalized state represents a specific burst equivalent set (see Proposition 3.8 below for a formal statement of this fact). That being so, a generalized realization serves the important purpose of creating a deterministic relationship between output values and generalized states of a possibly non-deterministic machine. An examination of Definitions 3.3 and 3.4 leads to the following algorithm for the construction of generalized realizations (see also [13, 14]).

Algorithm 3.5: *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 $(r_1, v_1), (r_2, v_2), \ldots, (r_\rho, v_\rho)$. Let χ be a set that is disjoint of 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 $S := \{(r_1,v_1),(r_2,v_2),\ldots,(r_\rho,v_\rho)\}$ and the sets $\Upsilon := \varnothing$ and $\Upsilon' := \varnothing$. Assign $i := 1$.
Step 3. Let $\sigma_i$ be the i-th member of S, and let $\{G_1,\ldots,G_k\}$ be the family of burst equivalent subsets in $s(\sigma_i)$.
Step 4. Denote $\xi_j := \Phi(G_j), j = 1,2,\ldots,k$, and replace $\Upsilon$ by the set $\Upsilon \cup \{\xi_1,\ldots,\xi_k\}$. Assign $s_g(\sigma_i) := \{\xi_1,\ldots,\xi_k\}$, and denote $S(\xi_j) := G_j, j = 1,2,\ldots,k$.
Step 5. If $i + 1 \le \#S$, 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 $\Upsilon$ 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 $\Sigma$. The generalized output function is given by $h_g(\zeta) := h(S(\zeta))$ for all $\zeta \in \tilde{X}$. □
```

Example 3.6: Applying Algorithm 3.5 to the machine Σ of Example 2.5 leads to the addition of one group state $x^5 := \{x^2, x^3\}$; this yields the generalized state set $\tilde{X} = \{x^1, x^2, x^3, x^4, x^5\}$. The resulting generalized transition function s_g and output function h_g are described by the following table of transitions. □

Table 2. The generalized transitions of Σ and their outputs.

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

An asynchronous machine is said to have an infinite cycle if it has a valid pair (x, u) that has no next stable state.

Lemma 3.7: *If an asynchronous machine has no infinite cycles, neither does its generalized realization.*

Proof: Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalized realization of an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$. Assume, by contradiction, that Σ has no infinite cycles, while Σ_g does have an infinite cycle. Then, there is an input character $u \in A$ and a state $x_1 \in \tilde{X}$ for which Σ_g never reaches a stable combination. Let $x_1, x_2 \in s_g(x_1, u), x_3 \in s_g(x_2, u), \ldots$ be an infinite sequence of generalized states generated by the pair (x_1, u). Now, if all members of the sequence $x_1, x_2, x_3, \ldots$ are regular states of Σ, then Σ itself has an infinite cycle, contrary to our assumption. Thus, at least one of the states $\{x_1, x_2, \ldots\}$ must be a group state.

For a group state $x \in \tilde{X}$, denote by $S(x) \subset X$ the underlying set of states associated with x. By the definition of a generalized state, the machine Σ is at a (regular) state $x' \in S(x)$ when Σ_g is at the generalized state x. Build now the sequence $x'_1, x'_2, \ldots$ of regular states of Σ, where $x'_i := x_i$ when x_i is a regular state of Σ and $x'_i \in S(x_i)$ when x_i is a group state. But then, the infinite sequence $x'_1, x'_2, \ldots$ of regular states of Σ forms an infinite chain of transitions that does not lead to a stable state, contrary to our assumption that no such chains exist in Σ. This concludes the proof. □

For an asynchronous machine with critical races, it may not be possible to determine the exact state of the machine from its output burst, since different outcomes of a critical race may result in the same burst. The main feature of a generalized realization is that it resolves this uncertainty by creating a deterministic relationship between the output burst of a machine and the generalized state, as follows.

Proposition 3.8: *Let Σ be an asynchronous machine without infinite cycles. The generalized state of Σ is uniquely determined by the history of input values and output bursts of* Σ.

Proof: The proof is by induction. Consider a machine $\Sigma = (A, Y, X, x_0, f, h)$ with a generalized state set $\tilde{X}$ built in accordance with Algorithm 3.5. Initially, Σ and its generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ are both in a stable combination at the initial state x_0. Next, forming the induction assumption, consider the case where Σ is activated by an input string $u_0u_1 \dots u_p$ and, as a result, produces the string of bursts $\beta_0, \beta_1, \dots, \beta_p$. Assume that, from this information, it can be determined that Σ is at the generalized state $x \in \tilde{X}$.

Now, let u be an input character that forms a valid pair with the generalized state x, and let β be the burst generated by Σ as it moves to the next generalized stable state x' of the pair (x, u) (note that x' exists by Lemma 3.7). By induction, our proof will conclude upon showing that x' is uniquely determined by x, u, and β. Now, the next stable generalized state of Σ was reached through the burst β, so it must be a member of a burst equivalent class $S \subset s_g(x, u)$ associated with the burst β. By Definition 3.3, the class S is represented by exactly one generalized state, which, in this case, must then be the generalized state x'. Thus, x' is uniquely determined by x, u and β, and our proof concludes. □

4. Observers

Following the methodology employed by Geng and Hammer [4], the controller C of Fig. 1 is decomposed into a combination of two asynchronous machines: an observer ϑ and a control unit F, as described by Fig. 2.

Here, Σ is the asynchronous machine being controlled. The observer ϑ is an asynchronous machine that uses the input value and the burst of Σ to determine the generalized state of Σ. This generalized state is then used by the controller F—another asynchronous machine—to generate an input string that drives Σ to a desired outcome. The controller C of Fig. 1 is then the combination (F, ϑ).

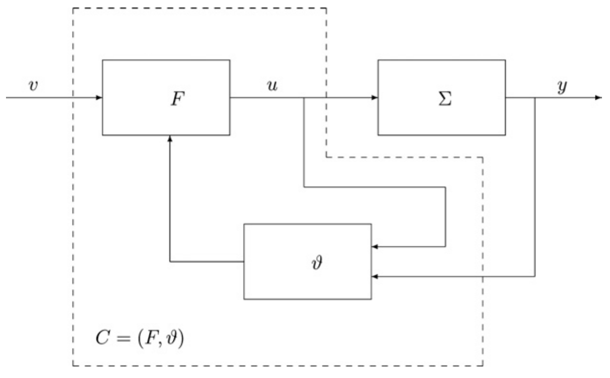

Fig. 2. Controller Structure.

The role of the observer ϑ here is similar to its role in [13, 14]: it determines whether Σ has reached a stable state, and it indicates the latest generalized stable state Σ has reached. The observer ϑ and the control unit F are designed to rest in a stable combination while Σ is in transition to guarantee fundamental mode operation of the configuration of Fig. 2.

The structure of the observer ϑ is determined by a generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ of Σ, as the observer's role is to reproduce each stable transitions of Σ_g immediately after it has occurred. Specifically, the observer is an input/state machine $\vartheta = (A \times Y^*, \tilde{X}, \tilde{X}, x_0, \sigma, I)$ with two inputs: the input character $u \in A$ of Σ and the output burst $\beta \in Y^*$ of Σ. The output set of ϑ and the state set of ϑ are identical to the generalized state set $\tilde{X}$ of the observed machine Σ. The initial state x_0 of ϑ is identical to the initial state of Σ.

The recursion function $\sigma : \tilde{X} \times A \times Y^* \to \tilde{X}$ of ϑ is constructed as follows. Assume that at step $i - 1$ the machine Σ is at the generalized stable combination (x, u'), when the input character changes to u, where (x, u) is a detectable pair. The change of the input character may give rise to a chain of transitions of Σ. Considering that the generalized realization has no infinite cycles (see Lemma 3.7), this chain of transitions consists of a finite number of steps, say steps $i, i+1, i+2, \dots, i+q$, and ends at the next stable state x'. Letting $y_i, y_{i+1}, \dots, y_{i+q}$ be the string of output values generated during this chain of transitions, we denote by $\beta := \beta(y_i, \dots, y_{i+q})$ the corresponding burst. As the machine Σ includes critical races, β is, in general, only one of the bursts that may result from this change of the input character; let $B(x, u)$ be the set of all such bursts, and let $\beta(x, u, x')$ be the subset of $B(x, u)$ that consists of all bursts created by transitions ending at the state x'.

Now, for a step $k \geq i$ along the resulting chain of transitions, let β_k be the part of the burst β from step i to step k. As the pair (x, u) was detectable, it is possible to determine when the burst terminates; at that point $k = i + q$ and $\beta_{i+q} \in \beta(x, u, x')$. These considerations lead us to the following definition of the recursion function σ of the observer ϑ:

$$\sigma(x, u, \beta_k) := \begin{cases} \zeta \in s_g(x, u) \text{ if } \beta_k \in \beta(x, u, \zeta), \\ x \text{ otherwise.} \end{cases} \tag{4.1}$$

For a detectable transition, Proposition 3.8 implies that the value of $\sigma(x, u, \beta_k)$ is uniquely determined by (4.1); non-detectable transitions must be avoided, since they do not permit fundamental mode operation of the closed

Table 3. The observer's transition function.

	$a,0$	$a,1$	$b,0$	$b,1$	$a,01$	$a,10$	$b,01$	$b,10$	$a,010$	$a,101$	$b,010$	$b,101$
x^1	–	–	x^1	–	–	–	–	–	–	–	–	–
x^2	x^2	–	–	–	–	–	–	–	–	–	x^1	–
x^3	x^3	–	–	–	–	–	–	–	–	–	x^1	–
x^4	–	x^4	–	–	–	–	–	x^1	–	–	–	–
x^5	x^5	–	–	–	–	–	–	–	–	–	–	–

loop configuration of Fig. 1. The observer ϑ is then an input/state asynchronous machine with the recursion

$$\vartheta: \ z_{k+1} = \sigma(z_k, u_k, \beta_k), k = 0, 1, 2, \ldots, \qquad (4.2)$$

where z_k is the current state of ϑ as well as the current output of ϑ.

In summary, the current state of the observer ϑ is equal to the last generalized stable state that the observed machine Σ has reached through a detectable transition; and ϑ switches to this state immediately following the step at which Σ reaches a stable combination at that state. Furthermore, this construction guarantees that ϑ remains in a stable combination while Σ is in transition, and that ϑ transits only while Σ is at a stable combination. In this way, ϑ helps create an environment within which Σ can be controlled in fundamental mode operation. The following example demonstrates the construction of an observer.

Example 4.1: We construct an observer for the generalized realization Σ_g derived in Example 3.6. Recall that the output alphabet of Σ_g is comprised of the two characters 0 and 1. An examination of the transition table of Example 3.6 shows that the longest burst generated by Σ_g during a one step stable transition consists of three characters and that the transition table representing the recursion function σ of the observer is as in Table 3 (Only stable and detectable transitions are listed; a nypnen in an entry indicates that the corresponding combination is not used.) □

5. The Control Unit

The construction of the control unit F of Fig. 2 is similar to its construction in [4], except for two points: (i) the construction is based on the generalized realization of Σ (Section 3) rather than on the original realization of Σ; and (ii) the notion of detectability of Definition 2.3 is used instead of the notion used in [4]. In this section, we outline the process that underlies the construction of F. We start in subsection 5.1 by generalizing the notion of stable reachability matrix to characterize all stable transitions of a generalized realization. Then, in subsection 5.2, we revisit the notion of feedback paths to help characterize the existence of feedback controllers. The family of all transitions that can be implemented by output feedback control is characterized in subsection 5. 3 by the generalized skeleton matrix (introduced there). Finally, output feedback controllers that utilize bursts to achieve model matching are presented in subsection 5. 4 3

5.1. The Generalized Reachability Matrix

Definition 5.1: Let Σ be an asynchronous machine with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ and denote $q := \#\tilde{X}$. For a pair of generalized states $x^i, x^j \in \tilde{X}$, define the set of input characters

$$\alpha(x^i, x^j) := \{a \in A : (x^i, a) \text{ is a detectable pair and } x^j \in s_g(x^i, a)\}. \qquad (5.1)$$

Then, the *generalized one-step reachability matrix* $R_g(\Sigma)$ is a $q \times q$ matrix whose i, j entry is given by

$$R_{g_{ij}}(\Sigma) := \begin{cases} \alpha(x^i, x^j) & \text{if } \alpha(x^i, x^j) \neq \varnothing, \\ N & \text{if } \alpha(x^i, x^j) = \varnothing, \end{cases} \qquad (5.2)$$

where $i, j = 1, 2, \ldots, q$, and N is a character not in A. □

Example 5.2: We construct the generalized one-step reachability matrix of the machine Σ of Example 2.5, using the generalized realization of Example 3.6. Note that the generalized one-step reachability matrix includes only one-step stable transitions that are detectable.

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

Next, we review a few operations on strings of characters ([12]) that are needed in order to derive features of the machine Σ from its generalized one-step reachability matrix. Consider two elements $w_1, w_2 \in A^+ \cup N$, where N is a character not in A. The operation $\uplus$ of *unison* is defined by

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

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

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

The *concatenation* of two members $w_1, w_2 \in A^+ \cup N$ 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

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

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

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

With this product, we can define the powers

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

Entry i,j of the power $R_g^t(\Sigma)$ consists of all strings of t characters that take Σ from a stable combination with the generalized state x^i to a stable combination with the generalized state x^j through a string of stable and detectable transitions; here, x^j could be just one possible outcome of a critical race.

We combine the powers of $R_g(\Sigma)$ into the matrix

$$R_g^{(t)}(\Sigma) := \uplus_{r=1,\ldots,t} (R_g(\Sigma))^r, t = 2, 3, \ldots \qquad (5.3)$$

Then, entry i,j of the matrix $R_g^{(t)}(\Sigma)$ consists of all strings of t or fewer characters that take Σ from a stable combination with the generalized state x^i to a stable combination with the generalized state x^j through a string of stable and detectable transitions; here as well, x^j could be just one possible outcome of a critical race.

Definition 5.3: The *generalized stable reachability matrix* of Σ is $\Gamma_g(\Sigma) := R_g^{(q-1)}(\Sigma)$, where q is the number of generalized states of Σ. □

The following statement, whose proof is similar to the proof of Murphy, Geng and Hammer [12, Lemma 3.2], demonstrates the significance of the generalized stable reachability matrix: it characterizes all pairs of states that are connected through strings of stable and detectable transitions.

Lemma 5.4: *Let Σ be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, \ldots, x^q\}$, and let $\Gamma_g(\Sigma)$ be the generalized stable reachability matrix of Σ. Then the following two statements are equivalent for all pairs of states $x^i, x^j \in \tilde{X}$.*

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

(ii) The (i,j) entry of $\Gamma_g(\Sigma)$ is not N. □

Example 5.5: The generalized stable reachability matrix of the machine Σ of Example 2.5 can be calculated by combining powers 1, 2, 3, and 4 of the one-step reachability matrix of Example 5.2. It is given by

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

5.2. Feedback Paths

A *deterministic transition* of an asynchronous machine Σ is a transition or a string of transitions with a unique outcome. The following notion helps characterize the set of all deterministic transitions possible for the closed loop machine of Fig. 1 (compare to [18–20]). Below, $\Pi_x : \tilde{X} \times A \to \tilde{X} : \Pi_x(x, u) := x$ denotes the standard projection onto the generalized state set.

Definition 5.6: Let Σ be an asynchronous machine with the generalized state set $\tilde{X}$ and the generalized stable recursion function s_g, and let x', x'' be two generalized states of Σ. A *detectable feedback path* from x' to x'' is a list $S_0, S_1, \ldots, S_p \subset \tilde{X} \times A$ of sets of detectable pairs with the following features:

(i) $S_0 = \{(x', u_0)\}$ for some $u_0 \in A$ (i.e., S_0 consists of a single pair);

(ii) $s_g[S_i] \subset \Pi_x[S_{i+1}], i = 0, \ldots, p-1$; and

(iii) $S_p = \{(x'', u_p)\}$, where $(x'', u_p) \in \tilde{X} \times A$ is a stable combination. □

The significance of a detectable feedback path originates from the following statement.

Theorem 5.7: *Let Σ be an asynchronous machine with the generalized realization Σ_g, and let x' and x'' be* *generalized states of Σ. Then, the following are equivalent.*

(i) There is an output feedback controller C for which the closed loop machine Σ_{g_c} has a deterministic stable transition from x' to x'' in fundamental mode operation.
(ii) There is a detectable feedback path from x' to x''.

Regarding the proof of Theorem 5.7, the construction described below builds a feedback controller that satisfies condition *(i)* of the theorem when condition *(ii)* is valid. This proves one direction of the theorem. The converse direction can be proved by reversing the argument line of the construction (see also the proof of an analogous result in [20]).

Construction 5.8: *Let Σ be an asynchronous machine with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$. Let x' and x'' be two generalized states of Σ, and let $\{S_0, S_1, \ldots, S_p\}$ be a detectable feedback path from x' to x''. Assume that Σ is at a stable combination with the state x' reached through a detectable transition. The following process constructs an output feedback controller that takes Σ to a stable combination with x'' in fundamental mode operation.*

Let $W \subset A$ be the prescribed set of command characters that prompt the controller to initiate the transition: when a character of W appears at the command input v of Fig. 2, the controller starts to generate a string that takes Σ from a stable combination with x' to a stable combination with x''. For future reference, it is convenient to denote this controller by the symbol $C(x', x'', W)$.

The controller $C(x', x'', W)$ is a combination of an observer ϑ and a control unit $F(x', x'')$, as described in Fig. 2. The observer ϑ is the input/state system given by (4.2); its output character z is equal to the latest stable generalized state reached by Σ through a detectable transition. As the machine Σ is at the generalized state x' reached through a detectable transition, the output of ϑ at the start of the transition is $z = x'$.

The control unit is an asynchronous machine $F(x', x'') = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ with two inputs: the external command input $v \in A$ of Fig. 2 and the output $z \in \tilde{X}$ of the observer ϑ. Considering that the observer is given by (4.2), it only remains to construct the control unit $F(x', x'')$.

The control unit $F(x', x'')$ stays in its initial state ξ_0 until the observer ϑ indicates that Σ has reached a stable combination with the generalized state x'. At that point, $F(x', x'')$ switches to the state ξ_1, ready for a command input character $v \in W$. If such a command character is received, $F(x', x'')$ starts to create an input string that takes Σ to a stable combination with the generalized state x''. The transition function ϕ and the output function η of $F(x', x'')$ are built as follows. Let $U(x')$ be the set of input characters that form stable combinations with the generalized state x'; then, assign

$$\phi(\xi_0, (z, t)) := \xi_0 \textit{ for all } (z, t) \in \tilde{X} \times A \setminus \{x'\} \times U(x'),$$
$$\phi(\xi_0, (x', u)) := \xi_1 \textit{ for all } u \in U(x'),$$
$$\eta(\xi_0, (z, u)) := u \textit{ for all } (z, u) \in \tilde{X} \times A.$$

Note that the control unit $F(x', x'')$ is transparent in the state ξ_0—it applies to Σ an input character equal to the command input character u it receives.

While in the state ξ_1, the control unit $F(x', x'')$ is transparent for all external input characters, except for characters $v \in W$, as these initiate the required transition. Thus, we set

$$\eta(\xi_1, (x', u)) := u \textit{ for all } u \notin W.$$

For characters $v \in W$, the control unit applies to Σ one of the characters of the set $U(x')$, say $u' \in U(x')$, so that Σ continues for now to stay in a stable combination with the state x' (to assure fundamental mode operation):

$$\eta(\xi_1, (x', v)) := u' \textit{ for all } v \in W.$$

Now, the control unit $F(x', x'')$ initiates an input string that ultimately takes Σ to a stable combination with the desired state x'' through a string of stable and detectable transitions. Recalling the detectable feedback path $\{S_0, S_1, \ldots, S_p\}$, let $S_0 := \{(x', u_1)\}$, where $u_1 \in A$. When the control unit $F(x', x'')$ receives an external input character $v \in W$ while in the state ξ_1, it moves to the state $\xi^1(x', W)$; upon reaching this state, it applies the input character u_1 to Σ:

$$\phi(\xi_1, (x', u)) := \xi_1 \textit{ for all } u \notin W,$$
$$\phi(\xi_1, (x', u)) := \xi^1(x', W) \textit{ for all } u \in W,$$
$$\eta(\xi^1(x', W)) := u_1.$$

By the definition of a detectable feedback path, the input character u_1 moves Σ to a stable combination with a generalized state $x_1 \in s_g(x', u_1)$ through a detectable transition. Following this detectable transition, the observer ϑ changes its output to x_1 right after Σ has reached x_1. Upon receiving the character x_1 from the observer, the control unit $F(x', x'')$ moves to the state $\xi^2(x', W)$:

$$\phi(\xi^1(x', W), (z, u)) := \xi^1(x', W)$$
$$\textit{for all } (z, u) \in \tilde{X} \times A \setminus \{s_g(x', u_1)\} \times W,$$
$$\phi(\xi^1(x', W), (x_1, v)) := \xi^2(x', W)$$
$$\textit{for all } x_1 \in s_g(x', u_1) \textit{ and } v \in W.$$

Next, select a pair $(x_1, u_2) \in S_1$. When $F(x', x'')$ reaches the state $\xi^2(x', W)$, we set the output function to apply the input character u_2 to Σ:

$$\eta(\xi^2(x', W)) := u_2.$$

This process continues in a step-by-step fashion. At the step r, where $1 \leq r \leq p-1$, the control unit $F(x', x'')$ is at the state $\xi^r(x', W)$. As we are following a detectable feedback path, all transitions are detectable; the observer ϑ indicates at step r that Σ is in a stable combination with the generalized state x_{r-1}, and $F(x', x'')$ applies the input character u_r to Σ, where $(x_{r-1}, u_r) \in S_{r-1}$. The input character u_r causes Σ to move to a generalized state $x_r \in s_g(x_{r-1}, u_r)$ through a detectable transition. When Σ arrives at x_r, the output of ϑ switches to x_r, and we set the control unit $F(x', x'')$ to move to the state $\xi^{r+1}(x', W)$:

$$\phi(\xi^r(x', W), (z, u)) := \xi^r(x', W)$$
$$\text{for all } (z, u) \in \tilde{X} \times A \setminus \{s_g(x_{r-1}, u_r)\} \times W,$$
$$\phi(\xi^r(x', W), (x_r, v)) := \xi^{r+1}(x', W)$$
$$\text{for all } x_r \in s_g(x_{r-1}, u_r) \text{ and } v \in W.$$

Next, select a pair $(x_r, u_{r+1}) \in S_r$. Upon reaching the state $\xi^{r+1}(x', W)$, the control unit $F(x', x'')$ applies to Σ the input character u_{r+1}:

$$\eta(\xi^{r+1}(x', W)) := u_{r+1}.$$

This makes Σ move to a generalized state $x_{r+1} \in s_g(x_r, u_{r+1})$ through a detectable transition. Being the last member of a detectable feedback path, S_p consists of a single stable combination with the generalized state x'' so that, at step p, Σ arrives at a stable combination with x'' through a detectable transition. This concludes the construction of the control unit $F(x', x'')$. The state set of $F(x', x'')$ is $\Xi(x', W) := \{\xi_0, \xi_1, \xi^1(x', W), \xi^2(x', W), \ldots, \xi^p(x', W)\}$. The resulting controller $C(x', x'', W) = (F(x', x''), \vartheta)$ satisfies condition (i) of Theorem 5.7. □

5.3. The Generalized Skeleton Matrix

The set of all pairs of generalized states that are connected by a detectable feedback path can be characterized by a matrix of zeros and ones called the *generalized skeleton matrix* (see [18, 19, 20, 13, 14]). This matrix is constructed by the following algorithm using the *meet* operation—a binary operation that may involve a string $a \in A^+$, the characters 0 and 1, and a character ω not included in the alphabet A; it is defined as follows.

$$0 \wedge 0 := 0;\ 0 \wedge 1 := 0;\ 1 \wedge 0 := 0;\ 1 \wedge 1 := 1;$$
$$0 \wedge a := 0;\ a \wedge 0 := 0;\ 1 \wedge a := \omega;\ a \wedge 1 := \omega.$$

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

Remark 5.9: In the algorithm listed below, the character ω is needed in the meet operation only when the controlled machine Σ has infinite cycles. As the machines considered here have no infinite cycles, the character ω can be ignored. □

The *length* of a string $a \in A^*$ is the number of characters of a.

Algorithm 5.10: *Let Σ be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, \ldots, x^q\}$, and let $\Gamma_g(\Sigma)$ be the generalized stable reachability matrix of Σ.*

```
Step 1. Replace all entries of $N$ in the matrix $\Gamma_g(\Sigma)$ by the number 0; denote the resulting matrix by $K^1$.
Step 2. Perform (a) below for each $i,j = 1, \ldots, q$; 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 row $i$, then perform the following operations:
        Delete any string included in $K^1_{ij}$ from all entries of row $i$ of the matrix $K^1$. Replace all resulting empty entries, if any, by the number 0. Replace entry $K^1_{ij}$ by the number 1.
    (b) Denote the resulting matrix by $K'(1)$. Delete from the matrix $K'(1)$ all strings of $A^+$ whose length is bigger than 1. Replace all empty entries, if any, by the number 0. Denote the resulting matrix by $K(1)$. Set $k := 1$.
Step 3. If $k \neq q$, then continue to Step 4. Otherwise, perform the following operations: If every entry of the matrix $K(q)$ is either 0 or 1, then set $K_g(\Sigma) := K(q)$ and terminate the Algorithm; if $K(q)$ includes characters other than 0 or 1, then set $k := 1$ and continue to Step 4.
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 repeat from Step 3 with the value $k+1$ for $k$. Otherwise, proceed to Step 5.
Step 5 Perform the following for every character $u \in A$ that appears in row $k$ of the matrix $K(k)$:
    (a) Denote by $j_1, j_2, \ldots, j_q$ the columns of row $k$ of $K(k)$ that include $u$. Let $J(u)$ be the meet of rows $j_1, j_2, \ldots, j_q$ 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.
    (c) If $J(u)$ has no entries of 1, then return to Step 4. Otherwise, continue to (d).
```

(d) If $J(u)$ has entries of 1, then let $i_1, \dots, i_r$ 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 column k of $K(k)$. In the matrix $K(k)$, perform the following operations on every row of $S(k)$:

(i) Delete from the row all occurrences of input characters that appear in columns $i_1, \dots, i_r$ of the row.
(ii) Replace columns $i_1, \dots, i_r$ of the row by the number 1.
(iii) If any entries of $K(k)$ remain empty, then replace them by the number 0. Return to Step 4. □

The matrix $K_g(\Sigma)$ obtained in Step 3 of Algorithm 5.10 is the *generalized skeleton matrix* of the machine Σ. Its significance is described in the following statement, whose proof parallels the proof of Venkatraman and Hammer [20, Proposition 7].

Proposition 5.11: *Let Σ be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, \dots, x^q\}$ and the generalized skeleton matrix $K_g(\Sigma)$. Then, the following two statements are equivalent for any $i,j \in \{1, \dots, q\}$.*

(a) There is a detectable feedback path from x^i to x^j.
(b) The (i,j) entry of $K_g(\Sigma)$ is 1.

Example 5.12: Applying Algorithm 5.10 to the stable reachability matrix of Example 5.5, we obtain the generalized skeleton matrix of the machine Σ of Example 2.5:

$$K_g(\Sigma) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \square$$

5.4. Model Matching

The solution of the model matching problem for asynchronous machines depends on the notion of *output equivalence lists* introduced in Geng and Hammer [2004 and 2005], which we review next.

Definition 5.13: Let $\Lambda = \{\Lambda^1, \dots, \Lambda^q\}$ and $W = \{W^1, \dots, W^q\}$ be two lists of subsets of a set $\tilde{X}$. The *length* of the list Λ is the number q of its members. The list W is a *subordinate list* of the list Λ (written $W \prec \Lambda$) if W has the same length as Λ and if $W^i \subset \Lambda^i$ for all $i = 1, \dots, q$. A list is *deficient* if it includes the empty set as a member. □

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 set $g^I(s)$ consists of all elements $\alpha \in S^1$ satisfying $g(\alpha) = s$.

Definition 5.14: Let Σ be an asynchronous machine with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$, and let $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be a stable-state asynchronous machine with the state set $X' = \{\zeta^1, \dots, \zeta^q\}$. The *generalized output equivalence list* of Σ with respect to Σ' is given by $E_g(\Sigma, \Sigma') := \{E^1, \dots, E^q\}$, where $E^i := h_g^I h'(\zeta^i), i = 1, \dots, q$. □

Note that member i of $E_g(\Sigma, \Sigma')$ consists of all generalized states of Σ that produce the same output value as state i of Σ'.

Example 5.15: Consider the machine Σ of Example 2.5 with the generalized realization of Example 3.6, and the model Σ' with the state set $X' = \{\zeta^1, \zeta^2\}$, the input alphabet $A = \{a, b\}$, the output alphabet $Y = \{0, 1\}$, and the transition function described by the following table:

Table 4. The model's transition function.

X'	a	b	Y
ζ^1	ζ^1	ζ^2	1
ζ^2	ζ^1	ζ^2	0

By observation, the generalized output equivalence list of Σ with respect to Σ' is then $E_g(\Sigma, \Sigma') = \{E^1, E^2\}$, where $E^1 = \{x^4\}$ and $E^2 = \{x^1, x^2, x^3, x^5\}$. □

The following adapts to our setting a notion introduced by Geng and Hammer [3, 4].

Definition 5.16: Let Σ be an asynchronous machine with the generalized state set $\tilde{X}$, and let Λ^1 and Λ^2 be two nonempty subsets of $\tilde{X}$. The *generalized reachability indicator* $r_g(\Sigma, \Lambda^1, \Lambda^2)$ is 1 if there is a detectable feedback path from every element of Λ^1 to an element of Λ^2; otherwise, $r_g(\Sigma, \Lambda^1, \Lambda^2) := 0$.

Let $\Lambda = \{\Lambda^1, \dots, \Lambda^q\}$ be a list of $q \geq 1$ non-empty subsets of $\tilde{X}$. The *generalized fused skeleton matrix* $\Delta_g(\Sigma, \Lambda)$ is a $q \times q$ matrix whose (i,j) entry is $\Delta_{g_{ij}}(\Sigma, \Lambda) := r_g(\Sigma, \Lambda^i, \Lambda^j), i,j = 1, 2, \dots, q.$ □

The significance of fused skeleton matrices originates from the following statement, whose proof parallels the proof of Geng and Hammer [4, Corollary 53]. (Skeleton matrices of deterministic asynchronous machines are discussed in [12].)

Theorem 5.17: *Let Σ be an asynchronous machine with the initial condition x_0. Let Σ' be a stably minimal asynchronous machine with the skeleton matrix $K(\Sigma')$, the state set $X' = \{\zeta^1, \ldots, \zeta^q\}$, and the initial condition $\zeta_0 = \zeta^d$, where $d \in \{1, 2, \ldots, q\}$. 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 generalized subordinate list $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ of the generalized output equivalence list $E_g(\Sigma, \Sigma')$ such that $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.

Moreover, when (ii) is valid, the controller C can be implemented as a combination of an observer ϑ and a control unit F as depicted in Fig. 2, with ϑ being given by (4.2).

As we can see from Theorem 5.17, model matching for an asynchronous machine depends on the derivation of an appropriate subordinate list of the output equivalence list. This can be accomplished by using the following variant of [4, Algorithm 54].

Algorithm 5.18: *Let Σ and Σ' be asynchronous machines, where Σ' is a stably minimal machine serving as a model. Let $E_g(\Sigma, \Sigma') = \{E^1, \ldots, E^q\}$ be the corresponding generalized output equivalence list, and let $K(\Sigma')$ be the skeleton matrix of Σ'. The following steps yield a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ \ldots \succ \Lambda(r)$ of generalized subordinate lists of $E_g(\Sigma, \Sigma')$. The members of the list $\Lambda(i)$ are denoted by $\Lambda^1(i), \ldots, \Lambda^q(i)$; they are subsets of the generalized state set $\tilde{X}$ of Σ.*

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

Recursion. *Assume that a subordinate list $\Lambda(k) = \{\Lambda^1(k), \ldots, \Lambda^q(k)\}$ of $E_g(\Sigma, \Sigma')$ has been constructed for an integer $k \geq 0$. For each pair of integers $i, j \in \{1, \ldots, q\}$, let $S_{ij}(k)$ be the set of all generalized states $x \in \Lambda^i(k)$ for which the (i,j) element of the generalized fused skeleton matrix $\Delta_g(\Sigma, \Lambda(k))$ is 0; i.e., $S_{ij}(k)$ consists of all $x \in \Lambda^i(k)$ from which there is no detectable feedback path ending at a generalized state belonging to $\Lambda^j(k)$. Then, denote*

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

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

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

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

Then, the next subordinate list is given by

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

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

In view of Theorem 5.17, a controller that solves the model matching problem exists if and only if Algorithm 5.18 yields a non-deficient subordinate list.

Example 5.19: Recalling the machines Σ and Σ' of Example 5.12, we apply Algorithm 5.18 to the output equivalence list of Example 5.15 (using the generalized skeleton matrix of Example 5.12). The Algorithm yields the subordinate list $\{\{\varnothing\}, \{x^4\}\}$, which is deficient. Hence, there is no controller that makes Σ_c stably equivalent to Σ' in fundamental mode operation. □

Once a non-deficient subordinate list of the output equivalence list is available, a controller that achieves model matching in fundamental mode operation can be built by the following construction, whose proof parallels the proof of Geng and Hammer [4, Theorem 51].

Construction 5.20: *Let Σ be an asynchronous machine with the generalized realization $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$, and let $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be a stably minimal asynchronous machine serving as a model. Assume that condition (ii) of Theorem 5.17 is valid, namely, there is a non-deficient subordinate list $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ of the generalized output equivalence list $E_g(\Sigma, \Sigma')$ for which $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$. Then, a controller C satisfying $\Sigma_c = \Sigma'$ can be assembled as a combination of the observer ϑ given by (4.2) and the control unit F constructed below.*

To describe the construction of F, assume that the model Σ' is at a stable combination (ζ^i, w') and that Σ is at a stable combination with a generalized state $x \in \Lambda^i$ it has reached through a detectable transition. As x was reached through a detectable transition, the output of the observer ϑ is x. Presently, the external command character switches from w' to w, causing Σ' to move to the stable state ζ^j; this implies that $K_{ij}(\Sigma') = 1$.

Considering that $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$, it follows that $\Delta_{g_{ij}}(\Sigma, \Lambda) = 1$; hence, there is a detectable feedback path that takes Σ from the state $x \in \Lambda^i$ to a stable combination with a state $x_r \in \Lambda^j$. By Theorem 5.7, it follows that there is a controller $C(x, x_r, w)$ that takes Σ through a deterministic transition from x to x_r in fundamental mode operation. The construction of $C(x, x_r, w)$ follows the steps described in Construction 5.8, where our current states x and x_r correspond to the states x' and x'' of Construction 5.8, respectively. This results in *a state-feedback control unit $F(x, x_r)$ with the state set $\Xi(x, \zeta^i, w) := \{\xi_0, \xi^1(x, \zeta^i, w), \ldots, \xi^r(x, \zeta^i, w)\}$, which provides the state feedback necessary to take the machine Σ through a deterministic transition from x to x_r in fundamental mode.*

To implement all transitions necessary for the model matching problem, we use the operation of controller join ([20]). The join $C := C(x, x_r, w) \vee C(x', x'_r, w')$ of two controllers $C(x, x_r, w)$ and $C(x', x'_r, w')$ is defined as follows: (i) C is equal to $C(x, x_r, w)$ when Σ is at the state x and the external command character switches to w; this takes Σ to a stable combination with x_r. (ii) C is equal to $C(x', x'_r, w')$ when Σ is at the state x' and the external command character switches to w'; this takes Σ to a stable combination with x'_r. (iii) C is transparent otherwise. Then, the controller C can be expressed by the join

$$C := \bigvee_{\substack{x \in \Lambda^i \\ w \in A \\ i=1,2,\ldots q}} C(x, x_r, w).$$

In this expression, the target state $x_r \in \tilde{X}$ is determined by the pair (x, w) in the following way: recalling that $x \in \Lambda^i$, let $\zeta^j = s'(\zeta^i, w)$ be the next stable state of the model Σ' induced by the pair (ζ^i, w); then, x_r is any stably reachable member of Λ^j.

Note that all controllers $C(x, x_r, w)$ share the same observer ϑ; letting F be the state-feedback control unit of the controller C and using the notation of Construction 5.8, it follows that the state set of F is given by

$$\Xi := \bigcup_{\substack{x \in \Lambda^i \\ w \in A \\ i=1,2,\ldots q}} \Xi(x, w).$$

The state set Ξ can often be reduced by using standard reduction techniques for asynchronous machines (e.g., [10]). □

Examples demonstrating a close analog of Construction 5.20 can be found in [4] and in [13].

6. Conclusion

We presented a methodology for the design of output feedback controllers that achieve deterministic model matching for asynchronous sequential machines with critical races. These controllers utilize output bursts to broaden the class of non-deterministic machines for which deterministic model matching can be achieved.

The controller design process depends in a critical way on the derivation of a generalized realization of the controlled machine. Once such a realization is obtained, techniques for the control of deterministic machines can be deployed to design controllers for non-deterministic machines.

References

1. Barrett G, Lafortune S. "Bisimulation, the supervisory control problem, and strong model matching for finite state machines," *Discrete Event Dynamic Systems: Theory and Application*, 1998; 8(4); 377–429.
2. Dibenedetto D, Saldanha A, Sangiovanni-Vincentelli A. "Model matching for finite state machines," *Proceedings of the IEEE Conference on Decision and Control*, December 1994, 3, 3117–3124.
3. Geng XJ, Hammer J. "Asynchronous sequential machines: input/output control," *Proceedings of the 12th Mediterranean Conference on Control and Automation*, Kusadasi, Turkey, June 2004.
4. Geng XJ, Hammer J. "Input/output control of asynchronous sequential machines," *IEEE Trans Automat Control*, 2005; 50(12): 1956–1970.
5. Hammer J. "On some control problems in molecular biology," *Proceedings of the IEEE conference on Decision and Control*, December 1994, 4098–4103.
6. Hammer J. "On the modeling and control of biological signal chains." *Proceedings of the IEEE conference on Decision and Control*, December 1995, 3747–3752.
7. Hammer J. "On the corrective control of sequential machines," *International J Control*, 1996a; 65(2): 249–276.
8. Hammer J. "On the control of incompletely described sequential machines," *International J Control*, 1996b; 63(6): 1005–1028.
9. Hammer J. "On the control of sequential machines with disturbances," *International J Control*, 1997; 67(3): 307–331.
10. Kohavi Z. "Switching and Finite Automata Theory," McGraw-Hill Book Company, New York, 1970.
11. Murphy TE, Geng XJ, Hammer J. "Controlling races in asynchronous sequential machines," *Proceeding of the IFAC World Congress*, Barcelona, July 2002.
12. Murphy TE, Geng XJ, Hammer J. "On the control of asynchronous machines with races," *IEEE Trans Automat Control*, 2003; 48(6): 1073–1081.
13. Peng J, Hammer J. "Input/Output Control of Asynchronous Sequential Machines with Races," *International J Control*, 2009a; 83(1): 125–144.
14. Peng J, Hammer J. "Generalized Realizations and Output Feedback Control of Asynchronous Sequential Machines with Races," *Proceedings of the European Control Conference*, Budapest, Hungary, August 2009.
15. Ramadge PJG, Wonham WM. "Supervisory control of a class of discrete event processes," *SIAM J Control Optimization*, 1987; 25(1): 206–230.
16. Thistle JG, Wonham WM. "Control of infinite behavior of finite automata," *SIAM J Control Optimization*, 1994; 32(4): 1075–1097.
17. Unger SH. "Hazards, critical races, and metastability," *IEEE Trans Computers*, 1995; 44(6): 754–768.
18. Venkatraman N, Hammer J. "Stable realizations of asynchronous sequential machines with infinite cycles,"

Proceedings of the 2006 Asian Control Conference, 2006a, 45–51, Bali, Indonesia.

19. Venkatraman N, Hammer J. “Controllers for asynchronous machines with infinite cycles,” *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, 2006b, 1002–1007, Kyoto, Japan.
20. Venkatraman N, Hammer J. “On the control of asynchronous sequential machines with infinite cycles,” *International J Control*, 2006c; 79(7): 764–785.
21. Yang JM, Hammer J. “State Feedback Control of Asynchronous Sequential Machines with Adversarial Inputs,” *International J Control*, 2008; 81(12): 1910–1929.