International Journal of Control
Vol. 83, No. 1, January 2010, 125–144

Input/output control of asynchronous sequential machines with races

Jun Peng and Jacob Hammer*

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

(*Received 23 September 2008; final version received 8 June 2009*)

The design of output feedback controllers that eliminate uncertainties caused by critical races in asynchronous sequential machines is considered. The objective is to build controllers that drive a race-afflicted machine so as to match a prescribed deterministic model. Necessary and sufficient conditions for the existence of such controllers are presented in terms of a numerical matrix derived from the given machine. When controllers exist, an algorithm for their construction is also provided. The discussion depends on the novel notion of 'generalised state', which helps represent the uncertainty created by critical races and facilitates the construction of controllers.

Keywords: asynchronous sequential machines; output feedback control; critical races

1. Introduction

Asynchronous sequential machines form a fast and efficient platform for the implementation of high-speed computing systems, and they play a critical role in the analysis and modelling of parallel computing systems and of processes in molecular biology (e.g. Hammer 1994). It is therefore of interest to develop control techniques that help overcome deficiencies and defects in the operation of such machines. A common defect in the operation of an asynchronous machine is a critical race – a flaw that causes the machine to exhibit unpredictable behaviour. Critical races may originate from malfunctions, design flaws, implementation flaws or from genetic flaws in biological systems (e.g. Hammer 1995).

The objective of the present article is to develop output feedback controllers that eliminate the effects of critical races on asynchronous sequential machines, creating closed-loop machines that are deterministic and match prescribed models. The use of feedback controllers allows us to restore proper operation of malfunctioning asynchronous machines without having to replace them. This is particularly important in applications where the defective machine cannot be replaced, as is the case with remote, inaccessible or biological systems. In addition, the use of corrective feedback controllers is often less costly than the replacement of an entire defective machine. The use of an output feedback controller is described in Figure 1. In the figure, Σ is the defective asynchronous machine being controlled, and C is another asynchronous machine that serves as an output feedback controller. The closed-loop machine is denoted by Σ_c. Our objective is to find a controller C for which Σ_c exhibits desirable behaviour.

The desirable behaviour of the closed-loop machine Σ_c is represented by a deterministic asynchronous machine Σ', which serves as a *model*. Our objective is to design an output feedback controller C for which Σ_c simulates Σ'. In particular, as Σ' is deterministic, such a controller C eliminates the effects of uncertainties caused by the critical races of Σ. The problem of designing the controller C is referred to as the *model matching problem*. In this article, we present necessary and sufficient conditions for the existence of model matching output feedback controllers; whenever such controllers exist, we provide an algorithm for their design.

We assume that a description of the defective machine Σ is provided. This is often the case in applications; for example, consider a molecular biology system modelled by an asynchronous sequential machine (e.g. Hammer 1995). When a malfunction occurs in such a system, a diagnostic process is conducted to characterise the fault that causes the malfunction. The description of the defective machine is then known and can be used in the construction of a corrective biochemical controller. Being part of a live organism, the faulty system cannot be replaced in this case; the only option is to use a corrective controller.

As another example, consider a mass produced device in which a malfunction was detected after production. A sample of the device can be analysed to characterise the fault responsible for the malfunction.

*Corresponding author. Email: hammer@mst.ufl.edu

ISSN 0020–7179 print/ISSN 1366–5820 online
© 2010 Taylor & Francis
DOI: 10.1080/00207170903104182
http://www.informaworld.com

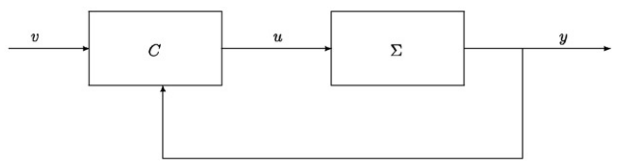


Figure 1. A feedback control configuration.

It is then often more economical to add a corrective controller to each faulty device than to completely replace all defective devices.

The problem of controlling asynchronous sequential machines was also discussed in Murphy, Geng and Hammer (2002, 2003), where state feedback controllers were used to eliminate the effects of critical races in asynchronous machines; in Venkatraman and Hammer (2006a, b, c), where state feedback controllers were used to eliminate the effects of infinite cycles; in Geng and Hammer (2005), where model matching with output feedback was considered for asynchronous machines with no critical races; and in Yang and Hammer (2008a, b), where state feedback controllers were used to counteract the effects of adversarial inputs and disturbances on asynchronous sequential machines. The present article concentrates on the design of output feedback controllers that eliminate the effects of critical races on asynchronous sequential machines. The handling of critical races in cases where there is no access to the state of the afflicted machine requires new theoretical notions. In particular, we introduce the notion of 'generalised state' to represent the impact of uncertainties caused by critical races (Section 2). In somewhat oversimplified terms, a generalised state represents a group of states consisting of all possible outcomes of a critical race. Although the particular outcome of a critical race is uncertain, the set of all possible outcomes is, of course, deterministic. As discussed in Section 2, the use of generalised states makes it possible to handle critical races within a deterministic framework.

Unlike synchronous machines, which are driven by clock pulses, asynchronous machines are driven by changes of their input variables. An asynchronous machine may occupy a *stable state* or a *transient state*. A stable state is a state at which the machine lingers until a change occurs in one of its input variables. A transient state is a state through which the machine passes very quickly, ideally in zero time. An asynchronous machine may pass through several transient states on its way from one stable state to another. Alternatively, during an infinite cycle, a machine moves indefinitely among transient states without ever reaching a stable state. The present article concentrates on machines that have no infinite cycles.

To guarantee predictable behaviour of an asynchronous machine, care has to be exercised to keep its input constant while the machine is not in a stable state. Indeed, if an input change occurs while the machine is in transition, then, due to the quick and asynchronous succession of transient states, it is not possible to predict the state of the machine at the time of the input change. This may result in an unpredictable response, as the response of a machine generally depends on its state. To avoid such uncertainty, asynchronous machines are normally operated so as to prevent input changes while the machine is not in a stable state. When this precaution is taken, the machine is said to operate in *fundamental mode*. In this article, all asynchronous machines are operated in fundamental mode.

The string of output values that an asynchronous machine generates along its way from one stable state to the next is often called a *burst*. A burst is a rapidly progressing string of output characters which, ideally, has a duration of zero time. Control of asynchronous machines can be performed with or without the utilisation of bursts (Geng and Hammer 2004, 2005). The use of bursts facilitates more powerful controllers at the cost of somewhat higher controller complexity. Often, control objectives can be met without utilising bursts. In such cases, avoiding the use of bursts simplifies controller design and implementation. The present article concentrates on the existence and the design of controllers that do not utilise bursts. The utilisation of bursts is considered in a separate report.

Studies dealing with other aspects of the control of sequential machines can be found in Ramadge and Wonham (1987), Thistle and Wonham (1994), and Kumar, Nelvagal and Marcus (1997), where the theory of discrete event systems is investigated; and in Dibenedetto, Saldanha and Sangiovanni-Vincentelli (1994), Hammer (1994, 1995, 1996a, b, 1997), Barrett and Lafortune (1998), and Yevtushenko, Villa, Brayton, Petrenko and Sangiovanni-Vincentelli (2008), where issues related to control and model matching for sequential machines are considered. These discussions do not take into consideration specialised issues related to the function of asynchronous machines, such as the implications of stable states, transient states, and fundamental mode operation. The lack of such consideration may result in non-deterministic behaviour of asynchronous sequential machines.

The article is organised as follows. Generalised states and their use in characterising the control capabilities of asynchronous machines with critical races are introduced and studied in Sections 2 and 3. Section 4 deals with the existence and the design of output feedback controllers that solve the model matching problem. The article concludes in Section 5 with a comprehensive example.

2. Basics

2.1 *Asynchronous machines and stable state machines*

An asynchronous sequential machine is represented by a sextuple $\Sigma = (A, Y, X, x_0, f, h)$, where A is a finite *input alphabet*, Y is a finite *output alphabet*, X is a finite *state set*, $x_0 \in X$ is the initial state of the machine, and $f: A \times X \to X$ (the *recursion function*) and $h: X \to Y$ (the *output function*) are partial functions. The machine operates according to the following recursion:

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

where $x_k \in X$ is the state, $u_k \in A$ is the input value, and $y_k \in Y$ is the output value of the machine at step k. The step counter k advances by one upon any change of the input or of the state of Σ. We use here the Moore representation of the machine Σ, so the output function h depends only on the state.

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

$$x_1 = f(x, u), \quad x_2 = f(x_1, u), \ldots \tag{2}$$

which may or may not terminate. If this chain of transitions terminates, then the last state, say, x_q, satisfies $x_q = f(x_q, u)$, i.e. (x_q, u) is a stable combination of Σ. In such case, we refer to x_q as *the next stable state* of x with the input character u. If the chain (2) does not terminate, then it forms an *infinite cycle*. In this article, we restrict our attention to asynchronous machines with no infinite cycles. It is convenient to define iterations of the recursion function by setting

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

A valid pair $(r, v) \in X \times A$ is a *critical race* if its next stable state is not uniquely determined and may be one of several options. The options $r_1, r_2, \ldots, r_m \in X$ of the next stable state are called the *outcomes* of the critical race (r, v) (e.g. Kohavi 1970). A critical race turns the machine Σ into a non-deterministic machine. The recursion function f of a machine with critical races is multivalued, and the symbols x_{k+1} in (1) and $x_1, x_2, \ldots$ in (2) may represent sets of states.

State transitions of asynchronous machines occur very quickly – ideally, in zero time. Consequently, for an asynchronous machine, the behaviour observed by a user is determined by state/input pairs at which the machine lingers, i.e. by stable combinations. To characterise this behaviour, let (x, u) be a valid pair of an asynchronous machine Σ. Denote by x' the next stable state of x with the input u when (x, u) is not a critical race; otherwise, let $r_1, r_2, \ldots, r_m$ be the outcomes of the race. Then, the *stable recursion function s* of Σ is defined by

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

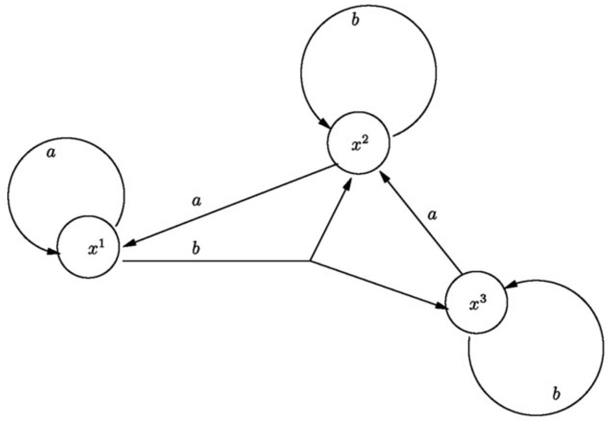
The *stable state machine* $\Sigma_{|s} = (A, Y, X, x_0, s, h)$ describes the stable transitions of Σ; it shares with Σ the input, output, and state sets, as well as the initial state and the output function. A stable state machine is *minimal* if there is no stable state machine with a smaller number of states that has the same input/output behaviour. Finally, two asynchronous machines Σ and Σ' are *stably equivalent* if the stable state machines $\Sigma_{|s}$ and $\Sigma'_{|s}$ are equivalent i.e. if $\Sigma_{|s}$ and $\Sigma'_{|s}$ have the same input/output behaviour. Stably equivalent machines are indistinguishable by a user, since they have the same stable combinations. We write $\Sigma = \Sigma'$ to indicate that Σ and Σ' are stably equivalent.

Example 2.1: Consider a machine Σ with the input alphabet $A = \{a, b\}$, the output alphabet $Y = \{0, 1\}$, the state set $X = \{x_1, x_2, x_3\}$, and the following transitions:

Table of transitions for Σ

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

Diagram of transitions for Σ

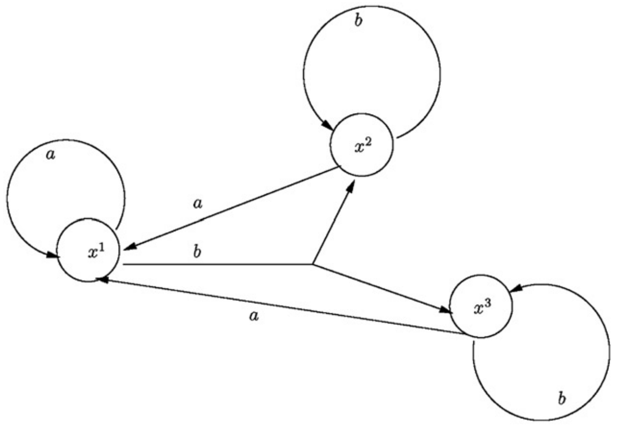


As we can see, the pair (x^1, b) is a critical race. The corresponding stable state machine $\Sigma_{|s}$ has the following transitions:

Table of transitions for $\Sigma_{|s}$

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

Diagram of transitions for $\Sigma_{|s}$



To guarantee that no uncertainties are introduced as a result of improper operation, all our asynchronous machines are operated in fundamental mode, namely, inputs of a machine are allowed to change only while the machine is in a stable combination (e.g. Kohavi 1970). In this way, the machine is in a well-defined state when an input change occurs. Fundamental mode operation is the most common mode of operating asynchronous sequential machines. For the configuration of Figure 1, fundamental mode operation translates into the following.

Condition 2.2: Let Σ and C be asynchronous sequential machines interconnected as in Figure 1. The configuration operates in fundamental mode when all the following hold:

(i) C is in a stable combination while Σ undergoes transitions;
(ii) Σ is in a stable combination while C undergoes transitions; and
(iii) the external input v changes only while Σ and C are both in a stable combination.

In order to implement part (i) of Condition 2.2, it must be possible for the controller C to determine whether the machine Σ has reached a stable combination, i.e. whether the transition process has terminated. As C has access only to the input and the output of Σ, this determination must be made without access to the state of Σ. The following notion is critical in this context.

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

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

In preparation for deriving necessary and sufficient conditions for strong detectability, consider an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ with the stable recursion function s. Assume that Σ is at a stable combination with the state x when the input character changes to u. This results in a chain of transitions $x = f^0(x, u), x^1 = f(x, u), x^2 = f^2(x, u), \ldots, x' = f^i(x, u)$, where $x' = s(x, u)$ is the next stable state of Σ (recall that our machines have no infinite cycles); the symbols $x, x^1, x^2, \ldots, x^i$, and x' may represent sets of states. Denoting the operation of set difference by \, the set of transient states included in this chain of transitions is given by

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

For a subset of states $S \subset X$ and an input value u that forms a valid combination with every state $x \in S$, denote by

$$f^{-}[S, u] := \bigcup_{x \in S} f^{-}(x, u) \tag{3}$$

the collection of all transient states included in transition chains triggered by the input character u from states of S. (Throughout the article, the symbol $\subset$ indicates inclusion with the possibility of equality.)

We can derive now a simple necessary and sufficient condition for strong detectability. Indeed, the set $h[f^{-}[S, u]]$ includes all output values that are generated by transient states, whereas the set $h[s[S, u]]$ includes all output values that are generated by stable states reached at the end of the transition process. In order for us to be able to determine whether transitions have ceased, these two sets must be disjoint. In fact, the converse is also true, and we obtain the following.

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

Proof: By (3), the set $h[f^{-}[S, u]]$ consists of all output characters generated by transient states that the machine Σ passes along its way from states of S to their next stable state. Now, assume first that $h[f^{-}[S, u]] \cap h[s[S, u]] \neq \emptyset$, so that there is an element $y \in h[f^{-}[S, u]] \cap h[s[S, u]]$. Then, there are states x',x'' of the machine Σ such that $x' \in f^{-}[S, u]$, $x'' \in s[S, u]$, and $h(x') = h(x'') = y$; note that x' is a transient state, while x'' is a stable state. Thus, when the output character y appears, it is impossible to tell whether Σ is in the transient state x' or in the stable state x''. Consequently, (S, u) is not strongly detectable.

Conversely, assume that $h[f^{-}[S, u]] \cap h[s[S, u]] = \emptyset$, and that an output value $y \in h[s[S, u]]$ is detected. Then, the last equality implies that $y \notin h[f^{-}[S, u]]$, so that Σ cannot be at a transient state. Thus, (S, u) is strongly detectable, and our proof concludes. □

Example 2.5: For the machine Σ of Example 2.1, let $S = \{x^1, x^3\}$ and consider the input character $u = a$. Then, we have $s[S, a] = \{x^1\}$, while $f^{-}[S, a] = \{x^2, x^3\}$, so that $h[s[S, a]] = \{0\}$ and $h[f^{-}[S, a]] = \{1\}$. Thus, $h[f^{-}[S, a]] \cap h[s[S, a]] = \emptyset$ in this case, and the pair $(\{x^1, x^3\}, a)$ is strongly detectable. Similarly, it can be seen that the following pairs are all strongly detectable: $(x^1, a), (x^1, b), (x^2, a), (x^2, b), (\{x^2, x^3\}, a), (\{x^2, x^3\}, b)$.

2.2 *Generalised states*

In the present subsection, we introduce one of the most fundamental notions of our discussion. First, some terminology. Two states x, x' of a machine $\Sigma = (A, Y, X, x_0, f, h)$ are *output equivalent* if they yield the same output values, namely, if $h(x) = h(x')$. Given a set S of states, we can induce an *output equivalence partition* $\{S_1, S_2, \ldots, S_p\}$, which consists of disjoint classes $S_1, S_2, \ldots, S_p$ of output equivalent states of S. As our information about Σ must be derived from its output values, output equivalent states play an important role in our discussion.

It is sometimes convenient to group several states of a machine into one entity. For example, when creating an output equivalence partition, several states are grouped together into one output equivalence class. As another example, consider the case where the machine Σ passes through a critical race whose outcomes are all output equivalent. In such case, it is impossible to determine the exact state of the machine after the race from input and output values. It is then convenient to group the set of all states that are consistent with the available data into one entity that represents the uncertainty about the state. This leads us to the following notion. (Below, $\#S$ denotes the cardinality of a set S, and $P(S)$ is the power set of S, i.e. the set of all subsets of S.)

Definition 2.6: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, let χ be a set disjoint from X and including at least $2^{\#X}$ elements and let $\Phi : P(X) \to X \cup \chi$ be an injective function satisfying $\Phi(x) = x$ for all states $x \in X$. With an output equivalent set $S \subset X$, associate the element $\xi := \Phi(S)$.

If $\#S > 1$, then ξ is called a *group state* of the machine 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) (or the pair (S, u)) is a *valid pair* if (x, u) is a valid pair for all $x \in S$.

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

We extend now recursion and output functions to group states. 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 generalised state set of Σ. For a member $\zeta \in \tilde{X}$, denote by $S(\zeta)$ the underlying set of states, where $S(x) := x$ for every state $x \in X$. Further, for a valid combination $(\zeta, u) \in \tilde{X} \times A$, let $\{S_1, \ldots, S_m\}$ be the output equivalence partition of the set $s[S(\zeta), u]$, and let $\zeta^i \in \tilde{X}$ be the generalised state associated with S_i, $i = 1, \ldots, m$. Then, the *generalised stable recursion function* $s_g : \tilde{X} \times A \to \tilde{X}$ is defined by

$$s_g(\zeta, u) := \{\zeta^1, \ldots, \zeta^m\}. \tag{4}$$

The set $\{\zeta^1, \ldots, \zeta^m\}$ may include group states and regular states. The *generalised output function* h_g: $\tilde{X} \to Y$ is

$$h_g(\zeta) := h[S(\zeta)] \quad \text{for all } \zeta \in \tilde{X}. \tag{5}$$

Note that, since $S(\zeta)$ is an output equivalence class, $h[S(\zeta)]$ is a single character of the output alphabet Y. The sextuple $\Sigma_g := (A, Y, \tilde{X}, x_0, s_g, h_g)$ forms a *generalised machine* associated with Σ. In view of its construction, the generalised machine has exactly the same input/output behaviour as the original machine Σ. In other words, Σ_g is simply a different realisation of Σ, and we often refer to Σ_g as a *generalised realisation* of Σ.

Note that, even after a critical race, the generalised state of a machine is always uniquely determined by the machine's output value, since each generalised state represents a known output equivalence class. That being so, a generalised realisation creates a deterministic relationship between output values and generalised states of a possibly non-deterministic machine. The following algorithm for the construction of generalised realisations is a consequence of Definition 2.6.

Algorithm 1: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous sequential machine with the stable recursion function s, and assume that Σ has ρ critical race pairs $(r_1, u_1), (r_2, u_2), \dots, (r_\rho, u_\rho)$. 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 generalised realisation $\Sigma_g := (A, Y, \tilde{X}, x_0, s_g, h_g)$ 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 $S := \{(r_1, u_1), (r_2, u_2), \dots, (r_\rho, u_\rho)\}$ and the sets $\Upsilon := \varnothing$ and $\Upsilon' := \varnothing$. Assign $i := 1$.
Step 3: Let σ_i be the i-th member of the family S, and let $\{G_1, \dots, G_k\}$ be the output equivalence partition of the set of states $s(\sigma_i)$.
Step 4: Let $\xi_j := \Phi(G_j)$, $j = 1, 2, \dots, k$, and replace Υ by the set $\Upsilon \cup \{\xi_1, \dots, \xi_k\}$. Assign $s_g(\sigma_i) := \{\xi_1, \dots, \xi_k\}$, and denote $S(\xi_j) := G_j$, $j = 1, 2, \dots, k$.
Step 5: If $i+1 \le \#S$, then replace i by $i+1$ and return to Step 3.
Step 6: Define the difference set $\Upsilon'' := [\Upsilon \backslash \Upsilon'] \setminus X$; then replace $\Upsilon' := \Upsilon$.
Step 7: If $\Upsilon'' = \varnothing$, then go to Step 9.
Step 8: Replace S by an ordered family consisting of all valid pairs $(S(\zeta), u)$, where $\zeta \in \Upsilon''$ and $u \in A$, and return to Step 3.
Step 9: Terminate the algorithm. The set Υ is the set of group states, $\tilde{X} := X \cup \Upsilon$ is the generalised state set, and s_g is the generalised stable recursion function of Σ.

Remark 1: Step 6 of Algorithm 1 singles out all new generalised states that arose during the current cycle of the algorithm. Step 8 prepares towards an extension of the generalised stable recursion function to all resulting new pairs.

Note that a generalised realisation is usually not a minimal realisation. Nevertheless, generalised realisations greatly simplify the process of designing controllers for asynchronous machines with critical races, since they create a deterministic relationship between the output value of a machine and its generalised state even in the aftermath of a critical race.

Example 2.7: Using Algorithm 1, we build a generalised realisation for the machine Σ of Example 2.1. In this case, Σ has only one critical race $\sigma_1 = (x^1, b)$; the race has two outcomes $s(\sigma_1) = s(x^1, b) = \{x^2, x^3\}$. As $h(x^2) = h(x^3) = 1$, the output equivalence partition of the set $s(\sigma_1)$ consists of the single class $G_1 = \{x^2, x^3\}$. Associating with G_1 the generalised state ξ_1, the generalised stable recursion function s_g is set to $s_g(\sigma_1) := \xi_1$. Further, $s_g(\xi_1, a) := s[G_1, a] = \{s(x^2, a), s(x^3, a)\} = \{x^1\}$, and $s_g(\xi_1, b) := s[G_1, b] = \{s(x^2, b), s(x^3, b)\} = \{x^2, x^3\} = \xi_1$. For the generalised output function, we have $h_g(\xi_1) := h(\{x^2, x^3\}) = 1$. At all other valid combinations, s_g is identical to s, and h_g is identical to h. Finally, replacing the symbol ξ_1 by the symbol x^4 for notational simplicity, we obtain the following generalised stable machine Σ_g:

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

We regard the computational complexity of Algorithm 1 as the number of pairs to which it extends the generalised stable recursion function outside the original state set. An asynchronous machine Σ is *simple* if any set of next stable states of a critical race's outcomes includes no output equivalent states. The following is a bound on the computational complexity of the algorithm.

Lemma 2.8: *Let Σ be a simple asynchronous machine with $\rho \ge 1$ critical races. Assume that each race has no more that p outcomes, and let m be the number of characters in the input alphabet of Σ. Then, the computational complexity of Algorithm 1 does not exceed ρpm.*

Proof: The fact that Σ has ρ critical races implies that, during the first ρ times that Algorithm 1 passes through Step 3, the sets $\{G_1, \dots, G_k\}$ are all outcomes of critical races. Considering that each critical race has no more than p outcomes, we conclude that that $k \le p$ in each one of the first ρ passes through Step 3 of the algorithm (i.e. the outcomes of one critical race cannot create more than p group states). Thus, the first ρ passes of Step 3 can create no more than a total of ρp group states. Using the fact that there are m input characters, it follows that the generalised stable recursion function must be extended over no more than ρpm pairs during the first ρ cycles of the algorithm. After the first ρ cycles, the sets $\{G_1, \dots, G_k\}$ of Step 3 are all sets of next stable states of critical race outcomes. As Σ is

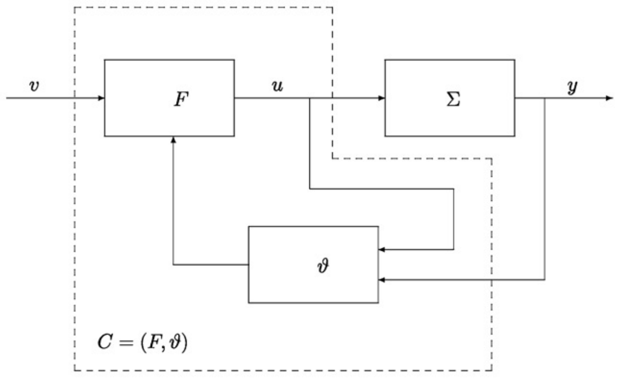


Figure 2. Controller decomposition.

a simple machine, it follows that $G_1, \ldots, G_k$ are all single states after the first ρ cycles of the algorithm, and hence give rise to no further group states. This concludes our proof. □

Next, we discuss the role of generalised realisations in the task of eliminating the effects of critical races.

3. Controlling asynchronous machines with races

Following Geng and Hammer (2004, 2005), we decompose the controller C of Figure 1 into two asynchronous machines: an observer ϑ and a control unit F as described in Figure 2.

As we discuss below, the observer ϑ determines the most recent generalised stable state visited by the machine Σ, while the control unit F generates the input strings that drive Σ. The overall controller C consists of the combination $C = (F, \vartheta)$.

3.1 *The observer* ϑ

The role of the observer in our discussion is analogous to its role in standard control theory: we use the observer to determine the most recent generalised stable state reached by the observed machine Σ. The nature of the observer here is somewhat different from its nature in Geng and Hammer (2004, 2005), as our present control process does not utilise bursts and has to overcome critical races of the controlled machine Σ.

In technical terms, let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalised machine induced by Σ. Then, the observer ϑ is a stable state input/state machine $\vartheta = (A \times Y, Z, z_0, \sigma)$ with two inputs: the input character $u \in A$ of Σ and the output character $y \in Y$ of Σ. The state set Z of ϑ consists of the same elements as the generalised state set $\tilde{X}$ of Σ, and the initial state of ϑ is identical to the initial state of Σ, i.e. $z_0 = x_0$. The recursion function $\sigma : Z \times A \times Y \to Z$ of ϑ is defined by

$$\sigma(z, u, y) := \begin{cases} \zeta \in s_g(z, u) & \text{if } y = h_g(\zeta) \text{ and } (z, u) \text{ is} \\ & \text{strongly detectable,} \\ z & \text{otherwise.} \end{cases} \tag{6}$$

In view of Proposition 2.4, the recursion function $\sigma(z, u, y)$ is well defined: when starting from a strongly detectable pair (z, u), the output value $y \in h_g(s_g(z, u))$ of Σ is reached only when Σ arrives at its next stable generalised state $\zeta \in s_g(z, u)$. Furthermore, by construction, there is exactly one generalised state $\zeta \in s_g(z, u)$ that satisfies $y = h_g(\zeta)$ for the current output value y of Σ. The state of the observer ϑ switches to the state ζ when it detects the character y in the signal it receives from the machine Σ. In this way, the state of the observer ϑ switches to ζ immediately after the machine Σ has reached its next stable generalised state. As a result, the state of ϑ tracks the stable generalised state of Σ, always lingering at the most recent stable generalised state that Σ has reached through a strongly detectable transition.

To illustrate the operation of the observer ϑ, assume that the machine Σ is at a stable combination (x, u') when the input character changes to u, where (x, u) is a strongly detectable pair. The change of the input character may give rise to a chain of transitions, ultimately leading Σ to a stable generalised state $\zeta \in s_g(x, u)$. As the pair (x, u) is strongly detectable, it follows by Proposition 2.4 that Σ starts displaying an output value $y \in h_g(s_g(x, u))$ right upon reaching the state $\zeta \in s_g(x, u)$, and not before. When the observer ϑ detects this output value, it transitions to its state $\zeta \in Z$, and, since ϑ is an input/state machine, this state becomes the new output value of ϑ.

The information provided by the observer ϑ helps guarantee fundamental mode operation of the control configuration 2 in the following way. As the state (and output) of ϑ stay constant while Σ is in transition, the control unit F has constant input while Σ is in transition. Hence, both ϑ and F (i.e. the entire controller C) remain in a stable combination while Σ is in transition (recall that the external input v is kept constant during transitions of the composite machine Σ_c). Now, immediately after Σ has reached its next stable combination through a strongly detectable transition, the observer ϑ undergoes a transition to its own next stable state. As ϑ is a stable input/state machine (see (6)), the output of ϑ remains constant until ϑ switches to its next stable state. When ϑ reaches its next stable state, its output changes and, as a result, the control unit F experiences a change at the input connected to ϑ. This, in turn, may lead F into a transition that changes the input u of the controlled machine Σ. Consequently, transitions among the three machines ϑ, F, and Σ always occur sequentially – one machine at a time. In this way, the closed-loop machine operates in fundamental mode.

3.2 *Reachability*

Needless to say, the presence of critical races in an asynchronous machine may create an uncertainty about the current state of the machine, since the output value does not allow one to distinguish among output equivalent states. As we have seen, the notion of generalised state helps us deal with this uncertainty by using a single generalised state to represent the set of all states that are compatible with available data. In the present subsection, we examine reachability properties within the set of generalised states. Our discussion in the previous subsection has shown that, for the closed-loop configuration of Figure 2, fundamental mode operation can be guaranteed only at strongly detectable pairs. As a result, the operation of the closed-loop must be restricted to such pairs, and this leads us to the following notion.

Definition 3.1: Let Σ be an asynchronous machine with the generalised realisation $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$, where $\tilde{X} = \{x^1, x^2, \ldots, x^\mu\}$. For a pair of generalised states $x^i, x^j \in \tilde{X}$, define the set of input characters

$$\alpha(x^i, x^j) := \{a \in A : (x^i, a) \text{ is a strongly detectable pair and } x^j \in s_g(x^i, a)\}. \quad (7)$$

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

$$R_{g_{ij}}(\Sigma) := \begin{cases} \alpha(x^i, x^j) & \text{if } \alpha(x^i, x^j) \neq \varnothing, \\ N & \text{if } \alpha(x^i, x^j) = \varnothing, \end{cases} \quad (8)$$

$i,j = 1, 2, \ldots, \mu$.

When the machine Σ has no group states, then the generalised one-step reachability matrix $R_g(\Sigma)$ reduces to the one-step reachability matrix of Murphy et al. (2002, 2003).

Let Σ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and the generalised state set $\tilde{X} = \{x^1, \ldots, x^n, x^{n+1}, \ldots, x^{n+t}\}$. Each group state x^{n+i}, $i = 1, 2, \ldots, t$, represents an underlying set of states that cannot be distinguished from each other by using input and output data. It is instructive to divide the one-step generalised reachability matrix into four blocks

$$R_g(\Sigma) = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix}$$

where R_{11} is an $n \times n$ matrix of strongly detectable one-step transitions among regular states of Σ; R_{12} is an $n \times t$ matrix of strongly detectable one-step transitions from regular states to group states, namely, strongly detectable transitions that pass through a critical race and have an uncertain outcome in terms of regular states; R_{21} is an $t \times n$ matrix of strongly detectable one-step transitions from a group state to a regular state, namely, transitions from an uncertain regular state to a single deterministic regular state; and R_{22} is a $t \times t$ matrix of one-step strongly detectable transitions from one group state to another group state, namely, transitions from an uncertain regular state that have uncertain outcomes in terms of regular states.

Example 3.2: For the machine Σ of Example 2.1, a generalised realisation was derived in Example 2.7 and a list of strongly detectable transitions is provided at the end of Example 2.5. The generalised one-step reachability matrix is then given by

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

The generalised one-step reachability matrix provides a simple characterisation of the generalised machine's critical races. Indeed, consider an asynchronous machine Σ with the stable recursion function s and the state set $X = \{x^1, \ldots, x^n\}$. Let $\tilde{X} = \{x^1, x^2, \ldots, x^{n+t}\}$ be the generalised state set and let s_g be the generalised stable recursion function of Σ. Assuming that Σ has critical races, there is a state x^i and an input value v for which the set $s(x^i, v)$ includes more than one state. If the states included in $s(x^i, v)$ are not all output equivalent, then the set $s_g(x^i, v)$ will also include more than one member. Then, by Definition 3.1, the input character v will appear in more than one entry of row i of $R_g(\Sigma)$. Thus, a critical race of the generalised machine causes an input character to appear more than once in a row of $R_g(\Sigma)$. A slight reflection shows that the converse is also true: if an input character v appears more than once in a row of $R_g(\Sigma)$, then v is involved in a critical race of the generalised machine. This leads to the following conclusion (compare with Venkatraman and Hammer 2006c, Proposition 4.16).

Lemma 3.3: *Let Σ be an asynchronous machine with the one-step generalised reachability matrix $R_g(\Sigma)$ and the generalised state set $\tilde{X} = \{x^1, \ldots, x^\mu\}$. Then, the following are equivalent for all $i = 1, \ldots, \mu$:*

(i) *An input character v appears in more than one entry of row i of $R_g(\Sigma)$.*

(ii) *The pair* (x^i, v) *is a critical race of the generalised realisation of* Σ.

Example 3.4: Consider the asynchronous machine Σ with the following table of transitions:

Table of transitions of Σ

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

Here, the outcome $\{x^2, x^3\}$ of the critical race splits into the two output equivalent subsets $\{x^2\}$ and $\{x^3\}$. As a result, this machine has no group states, and its generalised stable recursion function is identical to the original recursion function described in the table. The one-step generalised reachability matrix of Σ is then

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

As we can see, the input character b appears twice in row 1 of the matrix, signifying a critical race.

We define next a few operations on generalised one-step reachability matrices that are similar to the operations used in Venkatraman and Hammer (2006a, b, c). These operations allow us to characterise all reachability features of the machine Σ by applying an analog of matrix multiplication to the one-step reachability matrix.

Let A^+ be the set of all non-empty strings of characters of the alphabet A, and consider two elements $w_1, w_2 \in A^+ \cup N$, where N is a character not in A. Then, 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

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

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

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

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

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

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

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

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

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

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

By construction, the (i,j) entry of the matrix $R_g^q(\Sigma)$ is not N if and only if the generalised machine Σ_g can be taken from a stable combination with the generalised state x^i to a stable combination with the generalised state x^j in exactly q stable and strongly detectable transitions. Furthermore, if not N, this entry consists of all strings of q input characters that take Σ_g from a stable combination with x^i to a stable combination with x^j in exactly q stable and strongly detectable transitions.

In explicit terms, assume that the string of input characters $u_1u_2 \ldots u_q$ is a member of the (i,j) entry of $R_g^q(\Sigma)$. Then, we can proceed as follows from x^i to x_j: at a stable combination with the generalised state x^i, apply the input character u_1, and hold it until the observer ϑ of (6) displays a state of the set $s_g(x^i, u_1)$; then, apply the input character u_2, and again wait until the observer ϑ displays a state of the set $s_g(x^i, u_1u_2)$; and so on for q steps. If this string of transitions includes critical races, then x^j is one of the final state options.

For an integer $q \in \{1, 2, \ldots\}$, define the matrix

$$R_g^{(q)}(\Sigma) := \biguplus_{r=1,\ldots,q} R_g^r(\Sigma). \tag{9}$$

The (i,j) entry of $R_g^{(q)}(\Sigma)$, if not N, includes all input strings that may take Σ from a stable combination with the generalised state x^i to a stable combination with the generalised state x^j in q or fewer stable and strongly detectable steps. The following statement and its proof are similar to Murphy et al. (2003, Lemma 3.9).

Lemma 3.5: *Let* $\Sigma = (A, Y, X, x_0, f, h)$ *be an asynchronous machine with the generalised state set* $\tilde{X} = \{x^1, x^2, \ldots, x^\mu\}$, *and let* $R_g(\Sigma)$ *be the generalised* *one-step reachability matrix of* Σ. *Then, the following are equivalent*:

(i) *The generalised state* x^j *is stably reachable from the generalised state* x^i *through a finite string of stable and strongly detectable transitions, possibly as one outcome of a critical race.*
(ii) *The* (i,j) *entry of the matrix* $R_g^{(\mu-1)}(\Sigma)$ *is not* N.

In the lemma, when the transition forms a critical race, x^j is just one of the possible outcomes of the race and may not be deterministically reachable. In view of the lemma, all strongly detectable stable transitions of the generalised realisation of Σ are characterised by the matrix $R_g^{(\mu-1)}(\Sigma)$. This motivates the following notion.

Definition 3.6: Let Σ be an asynchronous machine having μ generalised states. The *generalised stable reachability matrix* of the machine Σ is $\Gamma(\Sigma) := R_g^{(\mu-1)}(\Sigma)$.

Example 3.7: Consider the machine Σ of Example 2.1. Here, the number of states in the generalised realisation is $\mu=4$, so $\mu-1=3$. Using the one-step generalised reachability matrix $R_g(\Sigma)$ of Example 3.2, we obtain

$$R_g^2(\Sigma) = \begin{pmatrix} \{a,ba\} & N & N & \{ab,b\} \\ \{a,ba\} & b & N & ab \\ \{a,ba\} & N & b & ab \\ \{a,ba\} & N & N & \{ab,b\} \end{pmatrix},$$

$$R_g^3(\Sigma) = \begin{pmatrix} \{a,ba,aba\} & N & N & \{ab,bab,b\} \\ \{a,ba,aba\} & b & N & \{ab,bab\} \\ \{a,ba,aba\} & N & b & \{ab,bab\} \\ \{a,ba,aba\} & N & N & \{ab,bab,b\} \end{pmatrix} \text{ and}$$

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

3.3 *Output feedback trajectories*

Output feedback trajectories form a critical tool for the construction of controllers that eliminate the effects of critical races. We start with a formal statement of the model matching problem, referring to Figure 1.

Problem 3.8: Let Σ and Σ' be asynchronous sequential machines with the same input and output alphabets, and assume that Σ' is a stable state machine. Find necessary and sufficient conditions for the existence of a controller C for which the closed-loop machine Σ_c is stably equivalent to Σ'. When C exists, describe an algorithm for its construction.

Normally, the model Σ' is a deterministic machine, while Σ may not be deterministic. In such case, the closed-loop machine has a deterministic stable state behaviour $\Sigma_{c|s}$ despite critical races of Σ, and the controller C eliminates the effects of these critical races. This means that every transition from one stable state to another stable state of the closed-loop machine Σ_c is deterministic. However, the transient states that Σ_c passes while executing such a transition may not be deterministic – they may differ from one execution of the transition to another. Recall that transient states are inconspicuous and have no impact on user experience, since transitions occur very quickly (ideally, in zero time).

Definition 3.9: A stable state transition is *deterministic* if it always has the same stable state outcome, irrespective of transient states passed during the transition process.

To demonstrate how different transition chains can lead to the same stable outcome of the closed-loop machine Σ_c, consider the following case. Let s be the stable recursion function of the machine Σ, and let (x',u) be a strongly detectable critical race of $\Sigma=(A,Y,X,x_0,f,h)$ with two distinct state outcomes x^p and x^q, i.e. $s(x',u)=\{x^p,x^q\}$. Assume that x^p and x^q are not output equivalent, so that $y^p:=h(x^p)\neq y^q:=h(x^q)$. Further, let x'' be another state of the machine Σ, and assume that there are input characters u_1, u_2 such that $s(x^p,u_1)=x''$ and $s(x^q,u_2)=x''$. Under these conditions, we can build a feedback controller that induces a deterministic stable transition from the state x' to the state x'', as follows. Assume that the machine Σ is at a stable combination with the state x', when the input character v is applied. As (x',u) is strongly detectable, our feedback controller can determine when Σ reaches its next stable state. If that next stable state is x^p, then the feedback controller detects the output value y^p; it applies then the input value u_1 to Σ, leading Σ to the state x''. Similarly, if the next stable state is x^q, then the feedback controller detects the output value y^q and applies the input value u_2 to Σ, again leading Σ to the stable state x''. Thus, irrespective of the outcome of the critical race, the closed-loop machine ends at a stable combination with the state x''. The transition process, however, is different for each outcome of the race. In formal terms, this process is captured by a generalisation of the notion of feedback trajectory (Venkatraman and Hammer 2006a, b, c) discussed next.

In qualitative terms, a feedback trajectory represents a chain of transitions of the machine Σ that starts at a single state and ends at a single state. Each intermediate step of the chain includes all possible outcomes of the previous step, and thus may involve multiple states. Specifically, let $\tilde{X}$ be the generalised state set of the machine Σ. A feedback trajectory consists of a string of subsets of pairs $S_0, S_1, \ldots, S_p \subset \tilde{X} \times A$, with the following important feature: any next stable state of a member of S_i appears in a member of S_{i+1}, $i = 0, 1, \ldots, p-1$. In this way, a feedback trajectory represents a natural succession of transitions: starting from the pair $(x_0, u_0) \in S_0$, the next generalised stable state is $x_1 \in s_g(x_0, u_0)$. This state appears in a member of S_1, say in the pair $(x_1, u_1) \in S_1$. Applying the input character u_1, we obtain a state $x_2 \in s_g(x_1, u_1)$, which appears in a member of S_2. Continuing in this way creates a string of transitions that takes Σ from the state x_0 to a state x_p in S_p. Individual transitions along this string may involve critical races, but a continuation is offered for each possible outcome.

A feedback trajectory forms the basis for deriving a feedback controller, as follows. Being aware of the machine's generalised state, a feedback controller can apply to Σ an input character that is paired with the machine's state in one of the feedback trajectory's members, thus taking the machine to the next step of the feedback trajectory. Eventually, this process allows the feedback controller to induce a transition between the beginning state and end state of the feedback trajectory. Below, $\Pi_x : \tilde{X} \times A \to \tilde{X} : \Pi_x(x, u) := x$ is the standard projection onto the generalised state set.

Definition 3.10: Let Σ be an asynchronous machine with the generalised state set $\tilde{X}$ and the generalised stable recursion function s_g, and let x' and x'' be two generalised states of Σ. An *output feedback trajectory* from x' to x'' is a list of subsets $S_0, S_1, \ldots, S_p \subset \tilde{X} \times A$ of strongly detectable pairs with the following features:

(i) $S_0 = (x', u_0)$ for some $u_0 \in A$.
(ii) $s_g[S_i] \subset \Pi_x[S_{i+1}]$, $i = 0, \ldots, p-1$.
(iii) $s_g[S_p] = x''$.

Example 3.11: For the machine Σ of Example 2.1 (see also Examples 2.5, 2.7, 3.2 and 3.7), an output feedback trajectory from the state x^2 to the generalised state x^4 is $S_0 = \{(x^2, a)\}$, $S_1 = \{(x^1, b)\}$, $S_2 = \{(x^4, b)\}$.

The notion of output feedback trajectory makes it possible to state necessary and sufficient conditions for the existence of a feedback controller, as follows.

Theorem 3.12: *Let Σ be an asynchronous machine, and let x' and x'' be two generalised states of Σ. Then, the following two statements are equivalent*:

(i) *There is an output feedback controller that drives Σ through a deterministic stable transition from x' to x'' in fundamental mode operation.*
(ii) *There is an output feedback trajectory from x' to x''.*

Proof: Let $\Sigma_g = (A, Y, \tilde{X}, x_0, s_g, h_g)$ be a generalised realisation of Σ. Assume that Σ is at a stable combination with the state x', having reached it by a strongly detectable transition. Now, assume that (ii) is valid, and let $\{S_0, S_1, \ldots, S_p\}$ be an output feedback trajectory from x' to x''. We construct an output feedback controller C that takes Σ from a stable combination with x' to a stable combination with x''.

Referring to Figure 2, recall that C has two inputs: the output value y of Σ and the external command character v. The latter commands the controller to induce a transition of Σ from x' to x''. Let $W \subset A$ be the set of external input characters that command this transition. To clarify the various relations, we use the notation $C(x', x'', W)$ for the controller: a character of W applied at the command input v causes the controller to generate an input string that takes Σ through a string of stable and strongly detectable transitions starting at x' and always ending at x''.

As indicated in Figure 2, the controller $C(x', x'', W)$ is a combination of an observer ϑ and a control unit F. The observer $\vartheta = (A \times Y, \tilde{X}, Z, z_0, \sigma, I)$ is a stable state input/state machine with two inputs: the input $u \in A$ of Σ and the output $y \in Y$ of Σ. The state set Z of ϑ is identical to the generalised state set $\tilde{X}$ of Σ; the initial condition z_0 is identical to that of Σ, namely, $z_0 = x_0$; and the stable recursion function σ of ϑ is given by (6). Denoting by ω the output value of ϑ, it follows from (6) that ω is equal to the latest stable generalised state reached by Σ through a strongly detectable transition. Thus, at the start of our process, the output of ϑ is $\omega = x'$.

The control unit F is an asynchronous machine $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ with two inputs: the external command input $v \in A$ and the output $\omega \in \tilde{X}$ of the observer ϑ. Recalling the length p of our output feedback trajectory, it turns out from the construction of F described below that the state set of F consists of $p+2$ states, which we denote by $\Xi = \{\xi_0, \xi_1, \xi^1(x', W), \xi^2(x', W), \ldots, \xi^p(x', W)\}$. The control unit F stays in its initial state ξ_0 until the observer ϑ indicates that Σ has reached a stable combination with the generalised state x'. Then, F switches to the state ξ_1; in this state, F is ready for a command input character $v \in W$. The transition function ϕ of F is constructed as follows. Let $U(x')$ be the set of all input characters that form stable combinations with the generalised state x'; then, set

$$\phi(\xi_0, (z, v)) := \xi_0 \quad \text{for all } (z, v) \in \tilde{X} \times A \setminus \{x' \times U(x')\},$$
$$\phi(\xi_0, (x', v)) := \xi_1 \quad \text{for all } v \in U(x').$$

While in the state ξ_0, the control unit F is transparent – it applies to Σ an input character identical to the external command input character it receives. Thus, we set the output function of F as follows:

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

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

$$\eta(\xi_1, (x', v)) := v \quad \text{for all } v \notin W.$$

For a character $v \in W$, 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':

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

This assignment helps achieve fundamental mode operation of the combined system.

When an external command character $v \in W$ is received while F is at the state ξ_1, the control unit F starts to create an input string to drive Σ to a stable combination with the generalised state x''. To construct this string, we employ the given output feedback trajectory $\{S_0, S_1, \ldots, S_p\}$. Suppose $S_0 = \{(x', u_1)\}$, where $u_1 \in A$. Upon receiving the command character v, F moves from ξ_1 to the state $\xi^1(x', W)$; upon reaching this state, F applies the input character u_1 to Σ. This is accomplished by setting

$$\begin{aligned}&\phi(\xi_1, (x', v)) := \xi_1 \quad \text{for all } v \notin W,\\ &\phi(\xi_1, (x', v)) := \xi^1(x', W) \quad \text{for all } v \in W,\\ &\eta(\xi^1(x', W)) := u_1.\end{aligned}$$

By the definition of an output feedback trajectory, the input character u_1 moves Σ to a stable combination with a generalised state $x_1 \in s_g(x', u_1)$ through a strongly detectable transition. Thus, the observer ϑ outputs the character x_1 immediately after Σ has reached its next stable combination (x_1, u_1). Upon detecting the output x_1 of ϑ, the controller F moves to the state $\xi^2(x', W)$. To this end, set

$$\begin{aligned}\phi(\xi^1(x', W), (z, v)) &:= \xi^1(x', W) \quad \text{for all } (z, v) \in\\ &\qquad \tilde{X} \times A \setminus \{x_1 \times W\},\\ \phi(\xi^1(x', W), (x_1, v)) &:= \xi^2(x', W) \quad \text{for all } v \in W.\end{aligned}$$

By the definition of an output feedback trajectory, there is a pair $(x_1, u_2) \in S_1$. When F reaches the state $\xi^2(x', W)$, it needs to apply the input character u_2 to the machine Σ, so we set

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

This process continues in a step-by-step fashion. Consider the step r, where $1 \le r \le p-1$; denote by x_r the stable generalised state of Σ at this point and by $\xi^r(x', W)$ the state of the control unit F. Recalling that all transitions along an output feedback trajectory are strongly detectable by definition, the current stable combination of Σ at the generalised state x_r was reached through a strongly detectable transition, and hence the observer ϑ shows x_r as its output. When F detects the output x_r of ϑ, it moves to the state $\xi^{r+1}(x', W)$, as follows:

$$\begin{aligned}\phi(\xi^r(x', W), (z, v)) &:= \xi^r(x', W) \quad \text{for all } (z, v) \in\\ &\qquad \tilde{X} \times A \setminus \{x_r \times W\},\\ \phi(\xi^r(x', W), (x_r, v)) &:= \xi^{r+1}(x', W) \quad \text{for all } v \in W.\end{aligned}$$

Next, using again the definition of an output feedback trajectory, there is a pair $(x_r, u_{r+1}) \in S_r$. Upon reaching the state $\xi^{r+1}(x', W)$, the control unit F needs to apply to Σ the input character u_{r+1}; to this end, set

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

The character u_{r+1} makes Σ move to a stable combination with a generalised state $x_{r+1} \in s_g(x_r, u_{r+1})$ through a strongly detectable transition. After $p-1$ such steps, this process takes Σ to a stable combination with the generalised state x'' through a strongly detectable transition, as desired. After reaching x'', the control unit returns to its initial state ξ_0 when the command input character changes to a character outside W:

$$\begin{aligned}\phi(\xi^p(x', W), (x'', v)) &: = \xi^p(x', W) \quad \text{for } v \in W,\\ \phi(\xi^r(x', W), (x'', v)) &:= \xi_0 \quad \text{for all } v \notin W.\end{aligned}$$

This concludes the construction of the control unit F. Note that with this control unit, the closed-loop machine operates in fundamental mode.

To prove the converse direction, assume that condition (i) of the theorem is valid, and let F be a control unit that takes Σ from a stable combination with the generalised state x' to a stable combination with the generalised state x'' in fundamental mode operation. Note that only the initial generalised state x' and the final generalised state x'' in this process are deterministic; intermediate steps encountered along the way may involve critical races and hence may not be deterministic. Denote by p the largest number of consecutive input characters that F applies to Σ in this transition process. We build now an output feedback trajectory $\{S_0, S_1, \ldots, S_p\}$ as follows. Let u_1 be the first input character that F applies to Σ after receiving an external command character $v \in W$, and set $S_0 := (x', u_1)$. Next, define the set of generalised states $X_1 := s_g(x', u_1)$. For each member $x \in X_1$, let $u_2(x)$ be the next input value applied by F to the machine Σ. Define the set $S_1 := \{(x, u_2(x)) : x \in X_1\}$. Continuing with this process to step r, where $r \in \{1, 2, \dots, p-1\}$, set $X_{r+1} := s_g[S_r]$. For each generalised state $x \in X_{r+1}$, let $u_{r+2}(x)$ be the next input character that F applies to Σ. Define the set of pairs $S_{r+1} := \{(x, u_{r+2}(x)) : x \in X_{r+1}\}$. According to (*i*), the closed-loop machine operates in fundamental mode; this implies that all transitions encountered along the way are strongly detectable. Hence, the list of subsets $\{S_0, S_1, \dots, S_p\}$ forms an output feedback trajectory from x' to x'', and our proof concludes. □

Example 3.13: Consider the machine Σ with the generalised realisation of Example 2.7. Using the output feedback trajectory $S_0 = \{(x^2, a)\}$, $S_1 = \{(x^1, b)\}$, $S_2 = \{(x^4, b)\}$ of Example 3.11, we follow the proof of Theorem 3.12 to construct a corresponding control unit F. Referring to Figure 2, recall that $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ has two inputs: the external command input $v \in A$ and the output $\omega \in \tilde{X}$ of the observer ϑ. Assume that the set of external command characters W that activate F is the single character W in this case. Note that the length of the output feedback trajectory is $p = 2$ here, so F has 4 states; it's state set is $\Xi = \{\xi_0, \xi_1, \xi^1(x^2, W), \xi^2(x^2, W)\}$.

Now, the control unit F stays in its initial state ξ_0 until the observer ϑ indicates that Σ has reached a stable combination with the state x^2. Then, F switches to the state ξ_1, getting ready for the command input character W. In our case, the set of input characters of Σ that form stable combinations with the state x^2 is $U(x^2) = \{\mathrm{b}\}$. The transition function ϕ of F is then set to

$$\phi(\xi_0, (z, v)) := \xi_0 \quad \text{for all } (z, v) \in \tilde{X} \times A \setminus \{x^2 \times b\},$$
$$\phi(\xi_0, (x^2, b)) := \xi_1.$$

While in the state ξ_0, the control unit F applies to Σ the input character it receives, namely,

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

While in the state ξ_1, the control unit F is transparent for all external input characters, except for the character b. At b, the controller F applies to Σ a control input character of the set $U(x^2)$; as this set consists of the single character b here, we can set in this case

$$\eta(\xi_1, (x', v)) := v \quad \text{for all } v \in A.$$

When the external command character W is received while F is at the state ξ_1, the control unit F starts to create an input string that drives Σ to a stable combination with the final state x^4 of our output feedback trajectory. Considering that $S_0 = \{(x^2, a)\}$, we set

$$\phi(\xi_1, (x^2, v)) := \xi_1 \quad \text{for all } v \neq W,$$
$$\phi(\xi_1, (x^2, W)) := \xi^1(x^2, W),$$
$$\eta(\xi^1(x^2, W)) := a.$$

According to our output feedback trajectory, the input character a moves Σ to a stable combination with the state x^1 through a strongly detectable transition. As a result, the observer ϑ outputs the character x^1 immediately after Σ has reached its next stable state. Upon detecting the output x^1 of ϑ, the controller F moves to the state $\xi^2(x^2, W)$; to this end, set

$$\phi(\xi^1(x^2, W), (z, v)) := \xi^1(x^2, W) \quad \text{for all } (z, v) \in \tilde{X} \times A \setminus \{x^1 \times W\},$$
$$\phi(\xi^1(x^2, W), (x^1, W)) := \xi^2(x^2, W).$$

According to our output feedback trajectory, the next step of the transition string requires F to apply the control input character b to the machine Σ, so we assign

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

This input character then takes Σ to the desired generalised state x^4. In this example, we simply leave the machine Σ at this state, so we let F remain at its last state by setting

$$\phi(\xi^2(x^2, W), (z, v)) := \xi^2(x^2, W) \quad \text{for all } (z, v) \in \tilde{X} \times A.$$

The next objective is to characterise the set of all pairs of generalised states that can be connected by an output feedback trajectory. This characterisation will allow us to determine the class of all models that can be matched by controlling a given machine Σ. We start with the following operation of 'meet', which was introduced in Venkatraman and Hammer (2006a, b, c). In a forthcoming algorithm, the meet helps us determine whether or not a particular pair of states can be connected by an output feedback trajectory.

Definition 3.14: Let A be an alphabet and let γ be a character that is not 0 nor 1 and is not included in A. The *meet* is a binary operation $\wedge$ involving a string $a \in A^+$ and the characters 0 and 1, and is defined as follows:

$$0 \wedge 0 := 0; \quad 0 \wedge 1 := 0; \quad 1 \wedge 0 := 0; \quad 1 \wedge 1 := 1,$$
$$0 \wedge a := 0; \quad a \wedge 0 := 0; \quad 1 \wedge a := \gamma; \quad a \wedge 1 := \gamma.$$

The meet of two vectors is the vector of the meets of the corresponding components.

Remark 2: The special character γ of Definition 3.14 is, in fact, necessary only when dealing with systems that include infinite cycles. As the systems under consideration in the present article have no infinite cycles, the parts involving γ can be omitted from the definition.

The next algorithm is similar to Algorithm 4.22 of Venkatraman and Hammer (2006c). It builds a matrix of zeros and ones that characterises all pairs of generalised states of Σ that can be connected by an output feedback trajectory. In qualitative terms, the algorithm works on the generalised stable reachability matrix $\Gamma(\Sigma)$, which characterises all stable and detectable transitions of the asynchronous machine Σ. Initially, the algorithm singles out all transitions that involve no critical races. For transitions that involve critical races, the algorithm checks backwards from a potential end state to a potential beginning state in a step-by-step manner, to determine whether all outcomes of intermediate steps lead to the same final state.

Algorithm 2: Let Σ be a an asynchronous machine with the generalised state set $\tilde{X} = \{x^1, \ldots, x^\mu\}$, and let $\Gamma(\Sigma)$ be the generalised stable reachability matrix of Σ.

Step 1: Replace all entries of N in the matrix $\Gamma(\Sigma)$ by the number 0; denote the resulting matrix by K^1.
Step 2: Perform (a) below for each $i, j = 1, \ldots, \mu$; then continue to (b):

(a) If the entry 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 $k \neq \mu$, then continue to Step 4. Otherwise, perform the following operations: If every entry of the matrix $K(\mu)$ is either 0 or 1, then set $K_g(\Sigma) := K(\mu)$ and terminate the algorithm. If $K(\mu)$ includes characters other than 0 or 1, then rename $K(1) := K(\mu)$, 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 entries 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 $i_1, \ldots, i_r$ 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 entries $i_1, \ldots, i_r$ of the row.
  (ii) Replace entries $i_1, \ldots, i_r$ 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.

Definition 3.15: The outcome $K_g(\Sigma)$ of Algorithm 2 is called the *generalised skeleton matrix* of the machine Σ.

It was shown in Venkatraman and Hammer (2006c) that Algorithm 2 has polynomial computational complexity. The matrix $K_g(\Sigma)$ characterises all pairs of generalised states that can be connected by an output feedback trajectory, as follows.

Proposition 3.16: *Let Σ be an asynchronous machine with the generalised skeleton matrix $K_g(\Sigma)$, and let x^i and x^j be two generalised states of Σ. Then the following two statements are equivalent*:

(a) *There is an output feedback trajectory from x^i to x^j.*
(b) *The (i,j) entry of $K_g(\Sigma)$ is* 1.

The proof of Proposition 3.16 is similar to the proof of Proposition 4.2 of Venkatraman and Hammer (2006c), except for some minor simplifications that originate from the fact that our present machine Σ has no infinite cycles.

Example 3.17: We demonstrate Algorithm 2 on an asynchronous machine Σ with the generalised state set $X = \{x^1, x^2, x^3, x^4, x^5\}$, the input alphabet $A = \{a, b\}$, and the generalised stable transition function s_g

described by the following table of transitions. We assume that all transitions are detectable.

X	a	b
x^1	$\{x^2, x^3\}$	x^1
x^2	x^2	$\{x^4, x^5\}$
x^3	x^3	x^4
x^4	x^5	x^4
x^5	x^5	x^5

The matrix of detectable stable transitions is then

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

Following Steps 1 and 2 of Algorithm 2, we obtain the matrix

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

Referring to Step 5 of the algorithm, we can see that row 1 includes characters that are not 0 or 1 in entries 2 and 3. The meet of rows 2 and 3 of the matrix $K(1)$ is $J(a) = (0, 0, 0, 0, 1)$. As the entries of $J(a)$ are all either 0 or 1, we proceed to part (b) of Step 5, where we obtain that $S(1)$ is rows 1 and 5. Applying the remaining operations of Step 5, we obtain a matrix of zeros and ones, terminating the algorithm with

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

Example 3.18: Consider the machine Σ of Example 2.1; its generalised stable reachability matrix $\Gamma(\Sigma)$ was derived in Example 3.7. Applying Algorithm 2 to $\Gamma(\Sigma)$, we obtain the generalised skeleton matrix

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

The process of calculating $K_g(\Sigma)$ also yields the corresponding output feedback trajectories, as follows: $x^1 \to x^4$: $\{\{(x^1, b)\}, \{(x^4, b)\}\}$; $x^2 \to x^1$: $\{\{(x^2, a)\}, \{(x^1, a)\}\}$; $x^2 \to x^4$: $\{\{(x^2, a)\}, \{(x^1, b)\}, \{(x^4, b)\}\}$; $x^3 \to x^1$: $\{\{(x^3, a)\}, \{(x^1, a)\}\}$; $x^3 \to x^4$: $\{\{(x^3, a)\}, \{(x^1, b)\}, \{(x^4, b)\}\}$; and $x^4 \to x^1$: $\{\{(x^4, a)\}, \{(x^1, a)\}\}$.

4. Controller design

In the present section, we describe the construction of model matching controllers. We start with some notation. Let S_1 and S_2 be two sets, and let $P(S_1)$ be the power set of S_1 (i.e. the set of all subsets of S_1). For a function $g: S_1 \to S_2$, denote by $g^I: S_2 \to P(S_1)$ the inverse set function of g. Specifically, for an element $s' \in S_2$, the set $g^I(s')$ consists of all elements $s \in S_1$ satisfying $g(s) = s'$.

We turn now to the generalisation of several notions introduced in Geng and Hammer (2004, 2005), starting with the notion of output equivalence list. The output equivalence list characterises the states of the controlled machine Σ that produce the same output character as a specific state of the model Σ'.

Definition 4.1: Let Σ be an asynchronous machine with the generalised realisation $\Sigma_g = (A, Y, \tilde{X}, s_g, h_g)$ and let $\Sigma' = (A, Y, X', f', h')$ be an asynchronous machine with the state set $X' = \{\zeta^1, \ldots, \zeta^q\}$. The *generalised output equivalence list* of Σ with respect to Σ' is $E_g(\Sigma, \Sigma') := \{E^1, \ldots, E^q\}\}$, where $E^i := h_g^I h'(\zeta^i)$, $i = 1, \ldots, q$.

In the notation of the definition, all generalised states of Σ that belong to the set E^i produce the same output value as the state ζ^i of Σ'. In general, the members of a generalised output equivalence list are not always disjoint sets.

Example 4.2: We calculate the generalised output equivalence list of the machine Σ of Example 2.1 with respect to the model $\Sigma' = (A, Y, X', \zeta_0, f', h')$ described by the following table of transitions:

X'	a	b	Y
ζ^1	ζ^1	ζ^3	1
ζ^2	ζ^1	ζ^2	0
ζ^3	ζ^1	ζ^3	0

For the machine Σ, we use the generalised realisation Σ_g derived in Example 2.7. As we can see, the output character corresponding to the state ζ^1 of the model Σ'

is the character 1. According to the table of transitions provided in Example 2.7, the set of states of the generalised realisation Σ_g that produce the output character 1 is $\{x^2, x^3, x^4\}$; consequently, $E^1 = \{x^2, x^3, x^4\}$. Similarly, $E^2 = \{x^1\}$, and $E^3 = \{x^1\}$.

Definition 4.3: Let Σ be an asynchronous machine, and let Λ^1 and Λ^2 be two non-empty sets of generalised states of Σ. The *generalised reachability indicator* $r_g(\Sigma, \Lambda^1, \Lambda^2)$ is 1 if there is an output feedback trajectory from every member of Λ^1 to a member of Λ^2; otherwise, $r_g(\Sigma, \Lambda^1, \Lambda^2) := 0$.

The generalised reachability indicator can be computed from the generalised skeleton matrix.

Example 4.4: For the machine Σ whose generalised skeleton matrix $K_g(\Sigma)$ is given in Example 3.18, consider the two subsets of states $\Lambda^1 = \{x^1, x^2\}$ and $\Lambda^2 = \{x^2, x^4\}$. As entries (1, 4) and (2, 2) of $K_g(\Sigma)$ are 1, it follows that $r_g(\Sigma, \Lambda^1, \Lambda^2) = 1$.

Definition 4.5: Let Σ be an asynchronous machine, and let $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ be a list of non-empty sets of generalised states of Σ. The *generalised fused skeleton matrix* $\Delta_g(\Sigma, \Lambda)$ is a $q \times q$ matrix whose (i,j) entry is $\Delta_{g_{ij}}(\Sigma, \Lambda) := r_g(\Sigma, \Lambda^i, \Lambda^j)$, $i,j = 1, 2, \ldots, q$.

Example 4.6: Continuing with Example 4.4, let $\Lambda := \{\Lambda^1, \Lambda^2\}$. Then, a direct examination of $K_g(\Sigma)$ yields

$$\Delta_g(\Sigma, \Lambda) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Definition 4.7: Let Σ be an asynchronous machine and let $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ and $W = \{W^1, \ldots, W^q\}$ be two ordered lists of sets of generalised states of Σ. The *length* of the list Λ is the number q of its members. The list W is a *subordinate list* of Λ, denoted by $W \prec \Lambda$, if W has the same length as Λ and $W^i \subset \Lambda^i$ for all $i = 1, \ldots, q$. A list is *deficient* if one of its members is the empty set.

4.1 *Existence of controllers*

We turn now to the derivation of necessary and sufficient conditions for the existence of model matching controllers. Given two $p \times p$ numerical matrices A and B, the expression $A \geq B$ indicates that every entry of the matrix A is not less than the corresponding entry of the matrix B, i.e. $A_{ij} \geq B_{ij}$ for all $i,j = 1, \ldots, p$. The following statement is analogous to Lemma 49 of Geng and Hammer (2005) and has a similar proof. It provides a necessary condition for model matching which, as we show later, is sufficient as well.

Lemma 4.8: *Let $\Sigma = (A, Y, X, x_0, s, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be asynchronous machines, where Σ' is a minimal stable state machine with the state set $X' = \{\zeta^1, \ldots, \zeta^q\}$ and the initial condition $\zeta_0 = \zeta^d$. Assume that there is an output feedback controller C for which the closed-loop machine Σ_c is stably equivalent to Σ' and operates in fundamental mode. Then, there is a non-deficient subordinate list Λ of the generalised output equivalence list $E_g(\Sigma, \Sigma')$ for which $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.*

The following statement, which is the main result of the present section, shows that the conclusion of Lemma 4.8 is also a sufficient condition for the existence of model matching controllers.

Theorem 4.9: *Let Σ and Σ' be asynchronous machines, where Σ' is a minimal stable state machine with the state set $X' = \{\zeta^1, \ldots, \zeta^q\}$ and the initial condition $\zeta_0 = \zeta^d$, and let $K(\Sigma')$ be the skeleton matrix of Σ'. Then, the following two statements are equivalent:*

(i) *There is a controller C for which Σ_c and Σ' are stably equivalent, with Σ_c operating in fundamental mode.*
(ii) *There is a non-deficient subordinate list Λ of the generalised output equivalence list $E_g(\Sigma, \Sigma')$ for which $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.*

Furthermore, when (ii) holds, the controller C can be designed as a combination of an observer ϑ and a control unit F as depicted in Figure 2, with ϑ being given by (6).

Proof: Let $\Sigma = (A, Y, X, x_0, s, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be stable state realisations of the two machines. In view of Lemma 4.8, (i) implies (ii); thus, it only remains to show that (ii) implies (i). Assume then that (ii) is valid, and let $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ be a non-deficient subordinate list of $E_g(\Sigma, \Sigma')$ for which $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$. Using Λ, we build a controller C for which the closed-loop machine Σ_c is stably equivalent to Σ' and operates in fundamental mode. We build the controller C according to Figure 2, so C is a combination of an observer ϑ and a control unit F, with the observer ϑ being given by (6). Thus, our proof will conclude upon the construction of the control unit F, which is an asynchronous machine with two inputs: the external command input $v \in A$ and the output $\omega \in \tilde{X}$ of the observer ϑ. Accordingly, we can write $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$.

Let w' be the external command input character of the closed-loop machine Σ_c. Assume that the model Σ' is at a stable combination at its state ζ^i with the input character w', while Σ is at a stable combination at its generalised state $x \in \Lambda^i$ with the input character u'. Recalling that Λ is a subordinate list of the generalised output equivalence list $E_g(\Sigma, \Sigma')$, the inclusion $x \in \Lambda^i$

implies that the output character of Σ is the same as the output character of Σ'. Also, for the closed-loop machine to operate in fundamental mode, the state x must be the outcome of a stable and strongly detectable transition of Σ.

Assume now that the external input character switches from w' to the character $w \in A$. Such a switch causes the model Σ' to move to the next stable state $\zeta^j = s'(\zeta^i, w)$. The presence of this transition implies that $K_{ij}(\Sigma') = 1$; consequently, as $\Delta_g(\Sigma, \Lambda) \geq K(\Sigma')$, we have that $\Delta_{g_{ij}}(\Sigma, \Lambda) = 1$ as well. Hence, by the definition of $\Delta_g(\Sigma, \Lambda)$, the inclusion $x \in \Lambda^i$ entails that there is an output feedback trajectory from x to a member of Λ^j, say to the generalised state $x' \in \Lambda^j$. Then, following the process described in the proof of Theorem 3.12, build a feedback controller $C(x, x', w)$, which takes the machine Σ from the current stable combination with x to a stable combination with x' in fundamental mode operation, upon receiving the command character w. By the definition of an output equivalence list, the output character of the machine Σ at the end of this process will be $h(x') = h[\Lambda^j] = h'(\zeta^j)$, i.e. it will be equal to the output of the model Σ'.

Now, given two controllers $C(x^1, x^2, w^1)$ and $C(x^3, x^4, w^2)$ of the kind constructed in the proof of Theorem 3.12, define a combination controller $C^\vee := C(x^1, x^2, w^1) \vee C(x^3, x^4, w^2)$ which operates as follows: $C^\vee$ is equal to $C(x^1, x^2, w^1)$ at the pair (x^1, w^1) and is equal to $C(x^3, x^4, w^2)$ at the pair (x^3, w^2); at all other pairs, $C^\vee$ is transparent, applying to Σ the external command input character of the closed-loop machine. In this notation, a controller C that solves our model matching problem is given by the combination

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

According to the proof of Theorem 3.12, all controller components $C(x, s'(x, w), w)$ include the observer ϑ and operate in fundamental mode; consequently, the same is true for the controller C. This concludes our proof. □

The construction of the controller C described in the proof of Theorem 4.9 is based on the subordinate list Λ mentioned in condition (ii) of the Theorem. Such a subordinate list can be obtained by an algorithm that resembles Algorithm 54 of Geng and Hammer (2005). In qualitative terms, the algorithm operates as follows. Consider the generalised output equivalence list $E_g(\Sigma, \Sigma') := \{E^1, \ldots, E^q\}$ of the machine Σ with respect to the model Σ', where Σ' has the state set $\{\zeta^1, \zeta^2, \ldots, \zeta^q\}$. To simulate the response of Σ' when it occupies the state ζ^i, the machine Σ can be in any generalised state belonging to the member E^i of the generalised output equivalence list. Assume now that Σ' transitions to the state ζ^j; then, to produce the same output value, Σ must move to a state belonging to E^j. However, some states of E^i may not permit a transition to a state of E^j. Such states cannot be used when attempting to make the machine Σ simulate the behaviour of Σ', which includes a transition from ζ^i to ζ^j. The following algorithm eliminates all such unsuitable states from the output equivalence list, creating a subordinate list that is used in the solution of the model matching problem.

Algorithm 3: Let Σ and Σ' be asynchronous machines, where Σ' is a minimal stable state machine with the skeleton matrix $K(\Sigma')$, and let $E_g(\Sigma, \Sigma') = \{E^1, \ldots, E^q\}$ be the generalised output equivalence list of Σ with respect to Σ'. The following steps yield a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ \cdots \succ \Lambda(r)$ of subordinate lists of $E_g(\Sigma, \Sigma')$. The members of the list $\Lambda(i) = \{\Lambda^1(i), \ldots, \Lambda^q(i)\}$ are subsets of the generalised state set $\tilde{X}$ of Σ.

Start step: Set $k := 0$ and $\Lambda(0) := E_g(\Sigma, \Sigma')$.

Recursion step: 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 states $x \in \Lambda^i(k)$ from which there is no feedback trajectory ending at a state of $\Lambda^j(k)$. 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 \ to denote set difference, define the subsets

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

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

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

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

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

The following statement, which adapts Corollary 66 of Geng and Hammer (2005) to our present framework, indicates that Algorithm 3 generates an appropriate subordinate list, whenever such a list exists.

Theorem 4.10: *Let* $\Lambda(r)$ *be the subordinate list of* $E_g(\Sigma, \Sigma')$ *generated by Algorithm* 3. *Then, in the notation of Theorem* 4.9, *the following two statements* *are equivalent*:

(i) *There is a controller C for which Σ_c and Σ' are stably equivalent, with Σ_c operating in fundamental mode.*
(ii) *The list $\Lambda(r)$ is non-deficient and $x_0 \in \Lambda^d$.*

Furthermore, when (ii) *is valid, then $\Delta_g(\Sigma, \Lambda(r)) \geq K(\Sigma')$.*

The proof of Theorem 4.10 is analogous to the proof of Corollary 66 of Geng and Hammer (2005). By combining Algorithm 3 with the proof of Theorem 4.9, we obtain a constructive procedure for solving the model matching problem and overcoming the effects of critical races on input/output asynchronous machines. The essential step in this procedure is the derivation of a generalised realisation.

5. Example

Consider the problem of designing a controller for the machine Σ of Example 2.1 to match the model $\Sigma' = (A, Y, X', \zeta_0, f', h')$ of Example 4.2. Let $x_0 = x^2$ be the initial state of Σ, while $\zeta_0 = \zeta^1$ is the initial state of Σ'. The generalised stable machine Σ_g associated with Σ was calculated in Example 2.7, and the generalised skeleton matrix of Σ was derived in Example 3.18. The skeleton matrix of the model Σ' can be derived from the table of transitions of Example 4.2, and is given by

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

The generalised output equivalence list of Σ with respect to Σ' was derived in Example 4.2, and is given by $E_g(\Sigma, \Sigma') = \{E^1, E^2, E^3\}$, where $E^1 = \{x^2, x^3, x^4\}$, and $E^2 = \{x^1\}$, $E^3 = \{x^1\}$. Algorithm 3 then terminates at the first step, yielding the subordinate list $\Lambda(1) = \{\Lambda^1(1), \Lambda^2(1), \Lambda^3(1)\}$, where $\Lambda^1(1) = \{x^2, x^3, x^4\}$, $\Lambda^2(1) = \{x^1\}$, and $\Lambda^3(1) = \{x^1\}$. The generalised fused skeleton matrix can then be calculated and is given by

$$\Delta_g(\Sigma, \Lambda(1)) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

A direct examination shows that $\Delta_g(\Sigma, \Lambda(1)) \geq K(\Sigma')$; consequently, by Theorem 4.9, there is a controller C that solves the present model matching problem. We calculate now such a controller C.

Recall that C is a combination of an observer ϑ and a control unit F, as described in Figure 2. The structure of the observer ϑ is given by (6). The states of the observer ϑ are $\{x^1, x^2, x^3, x^4\}$, reflecting the generalised states of Σ; the initial state of ϑ is x^2, equal to the initial state of Σ. According to Example 2.5, all pairs of Σ_g are strongly detectable. Consequently, the transition table of ϑ is identical to the transition table of Σ_g given in Example 2.7. Thus, it only remains to construct the control unit F, which we do next.

The construction of F follows the process and the notation of the proof of Theorem 3.12, which indicates that $F = (A \times \tilde{X}, A, \Xi, \xi_0, \phi, \eta)$ has the state set $\{\xi_0, \xi_1, \xi^1(x^2, \zeta^1, b), \xi^1(x^1, \zeta^3, a), \xi^1(x^4, \zeta^1, b)\}$, where ξ_0 is the initial state of F. In our case, the action of the control unit starts from the initial state of Σ, so we can combine the state ξ_1 with the state ξ_0; thus, the state set of F is here

$$\Xi = \left\{\xi_0, \xi^1(x^2, \zeta^1, b), \xi^1(x^1, \zeta^3, a), \xi^1(x^4, \zeta^1, b)\right\}.$$

Now, as the model is in a stable combination at the initial state ζ^1, the external command input character must be a. For the machine Σ, we have $U(x^2) = \{b\}$, so F must generate the character b as input to Σ in the initial state. Thus,

$$\phi(\xi_0, (x^2, a)) := \xi_0, \quad \eta(\xi_0) := b.$$

Next, assume that the external command input character switches from a to b. This would make the model Σ' switch to the state ζ^3 with the output value $h'(\zeta^3) = 0$. The machine Σ must then be moved from x^2 to a generalised state of the subordinate list member $\Lambda^3(1) = \{x^1\}$. According to the transition table of Σ_g (Example 2.7), we have $s_g(x^2, a) = x^1$, so the control unit F must generate the character a as input to Σ:

$$\phi(\xi_0, (x^2, b)) = \xi^1(x^2, \zeta^1, b),$$

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

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

Further, assume that the model is at the stable combination (ζ^3, b) with Σ being at the stable combination (x^1, a), when the external input character switches to a. This causes the model to move to the state $s'(\zeta^3, a) = \zeta^1$. To simulate this transition, the system Σ must move from x^1 to a state of the set $\Lambda^1(1) = \{x^2, x^3, x^4\}$. An examination of the transition table of Σ_g (Example 2.7) shows that the only available option for Σ_g is a move from x^1 to the generalised state x^4; this requires an application of the input character b to Σ. Note that the resulting transition is a critical race: Σ will move to one of the states x^2 or x^3 – it does not matter which one of these states Σ actually reaches. To accomplish this transition, the recursion function of F is constructed as follows:

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

$$\phi(\xi^1(x^1,\zeta^3,a),(x^1,a)) = \xi^1(x^1,\zeta^3,a),$$

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

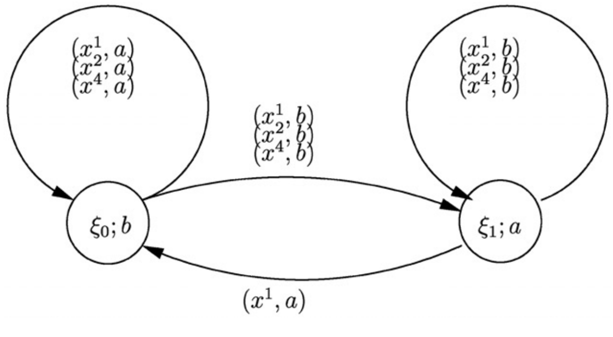
Continuing with our construction, consider the case where the model is at the stable combination (ζ^1, a) with Σ being at the stable combination (x^4, b), when the external input character switches to b. This causes the model to move to the state $s'(\zeta^1, b) = \zeta^3$. Therefore, the machine Σ must move to a state of the set $\Lambda^3(1)$, i.e. to the state x^1 in this case. According to the transition table of Σ_g (Example 2.7), F must generate a as the input character of Σ in order to accomplish this transition. To this end, we set

$$\phi(\xi^1(x^1,\zeta^3,a),(x^4,b)) = \xi^1(x^4,\zeta^1,b),$$

$$\phi(\xi^1(x^4,\zeta^1,b),(x^4,b)) = \xi^1(x^4,\zeta^1,b),$$

$$\eta(\xi^1(x^4,\zeta^1,b)) = a.$$

The recursion function and the output function of F are then completed in this manner. Note that some valid pairs of the controlled machine Σ may not be necessary to match the desired model; such pairs are not activated by the closed-loop machine Σ_c. Finally, employing standard techniques for the reduction of asynchronous machines (e.g. Kohavi 1970), the control unit F can be implemented using the following two state machine. In the diagram, the notation ξ_0; b indicates that the controller produces output b in state ξ_0, and ξ_1; a indicates that the controller produces the output a in state ξ_1.



To summarise, we have presented a constructive methodology for the design of output feedback controllers that solve the model matching problem for input/output asynchronous machines with critical races. The vital step in this methodology is the derivation of a generalised realisation of the controlled machine. The generalised realisation makes it possible to employ control techniques of deterministic asynchronous machines for the control of non-deterministic asynchronous machines.

References

Barrett, G., and Lafortune, S. (1998), 'Bisimulation, the Supervisory Control Problem, and Strong Model Matching for Finite State Machines', *Discrete Event Dynamic Systems: Theory and Application*, 8, 377–429.

Dibenedetto, M.D., Saldanha, A., and Sangiovanni-Vincentelli, A. (1994), 'Model Matching for Finite State Machines', *Proceedings of the IEEE Conference on Decision and Control*, 3, 3117–3124.

Geng, X.J., and Hammer, J. (2004), 'Asynchronous Sequential Machines: Input/output Control', in *Proceedings of the 12th Mediterranean Conference on Control and Automation*, Kusadasi, Turkey, June 2004.

Geng, X.J., and Hammer, J. (2005), 'Input/output Control of Asynchronous Sequential Machines', *IEEE Transactions on Automatic Control*, 50, 1956–1970.

Hammer, J. (1994), 'On Some Control Problems in Molecular Biology', in *Proceedings of the IEEE Conference on Decision and Control*, December 1994.

Hammer, J. (1995), 'On the Modelling and Control of Biological Signal Chains', in *Proceedings of the IEEE conference on Decision and Control*, December 1995.

Hammer, J. (1996a), 'On the Corrective Control of Sequential Machines', *International Journal of Control*, 65, 249–276.

Hammer, J. (1996b), 'On the Control of Incompletely Described Sequential Machines', *International Journal of Control*, 63, 1005–1028.

Hammer, J. (1997), 'On the Control of Sequential Machines with Disturbances', *International Journal of Control*, 67, 307–331.

Kohavi, Z. (1970), *Switching and Finite Automata Theory*, New York: McGraw-Hill Book Company.

Kumar, R., Nelvagal, S., and Marcus, S.I. (1997), 'A Discrete Event Systems Approach for Protocol Conversion', *Discrete Event Dynamic Systems: Theory & Applications*, 7, 295–315.

Murphy, T.E., Geng, X.J., and Hammer, J. (2002), 'Controlling Races in Asynchronous Sequential Machines', in *Proceeding of the IFAC World Congress*, Barcelona, July 2002.

Murphy, T.E., Geng, X.J., and Hammer, J. (2003), 'On the Control of Asynchronous Machines with Races', *IEEE Transactions on Automatic Control*, 48, 1073–1081.

Ramadge, P.J.G., and Wonham, W.M. (1987), 'Supervisory Control of a Class of Discrete Event Processes', *SIAM Journal of Control and Optimization*, 25, 206–230.

Thistle, J.G., and Wonham, W.M. (1994), 'Control of Infinite Behaviour of Finite Automata', *SIAM Journal on Control and Optimization*, 32, 1075–1097.

Venkatraman, N., and Hammer, J. (2006a), 'Stable Realisations of Asynchronous Sequential Machines with Infinite Cycles', in *Proceedings of the 2006 Asian Control Conference*, Bali, Indonesia.

Venkatraman, N., and Hammer, J. (2006b), 'Controllers for Asynchronous Machines with Infinite Cycles', in *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto, Japan.

Venkatraman, N., and Hammer, J. (2006c), 'On the Control of Asynchronous Sequential Machines with Infinite Cycles', *International Journal of Control*, 79, 764–785.

Yang, J.M., and Hammer, J. (2008a), 'State Feedback Control of Asynchronous Sequential Machines with Adversarial Inputs', *International Journal of Control*, 81, 1910–1929.

Yang, J.M., and Hammer, J. (2008b), 'Counteracting the Effects of Adversarial Inputs on Asynchronous Sequential Machines', in *Proceedings of the IFAC World Congress*, Seoul, Korea, July 2008.

Yevtushenko, N., Villa, T., Brayton, R., Petrenko, A., and Sangiovanni-Vincentelli, A. (2008), 'Compositionally Progressive Solutions of Synchronous {FSM} Equations', *Discrete Event Dynamic Systems*, 18, 51–89.