International Journal of Control
Vol. 79, No. 7, July 2006, 764–785

Taylor & Francis
Taylor & Francis Group

On the control of asynchronous sequential machines with infinite cycles

N. VENKATRAMAN† and J. HAMMER*‡

†Department of Electrical Engineering, Northern Arizona University, Flagstaff, AZ 86011, USA
‡Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA

(*Received 9 March 2005; in final form 28 February 2006*)

The problem of eliminating the effects of infinite cycles on asynchronous sequential machines is considered in a control theoretic context. The main objective is to develop state feedback controllers that stop infinite cycles in an existing asynchronous machine, while controlling the machine to match a prescribed model. Necessary and sufficient conditions for the existence of such controllers are derived in terms of an inequality condition between two numerical matrices. The results include an algorithm for the characterization of all infinite cycles of a given machine as well as an algorithm for the construction of appropriate controllers, whenever they exist.

1. Introduction

Asynchronous sequential machines, or, as they are sometimes called, clockless logic circuits, form the building blocks of some of the fastest computing machines. An infinite cycle is a common defect of an asynchronous sequential machine. It causes the machine to loop indefinitely among several of its states. Infinite cycles can occur as a result of malfunctions, design flaws, component failures, or implementation flaws. In this paper, we develop techniques to control an asynchronous machine so it meets the following two objectives: (i) the controlled machine does not linger in an infinite cycle; and (ii) the performance of the controlled machine matches a desired model. The basic control configuration is described by figure 1.

In figure 1, Σ is the asynchronous machine being controlled and C is another asynchronous machine that serves as a feedback controller. We denote by Σ_c the closed loop machine represented by the diagram. The controller C drives the machine Σ so as to eliminate the effects of infinite cycles and to match specified performance. In brief terms, we refer to C as a *corrective controller*.

A corrective controller can be designed so that the closed loop system will function properly whether or not the machine Σ is afflicted by an infinite cycle. In this way, the corrective controller can be used as a preemptive measure against malfunctions before they occur, improving system reliability. In addition, the use of a corrective controller is often economically more efficient than a complete replacement of a faulty system. It is the only practical solution in cases where the affected system is inaccessible.

The existence of a corrective controller depends on certain reachability properties of the faulty machine Σ. These properties can be characterized in terms of a numerical matrix of zeros and ones, called the "skeleton matrix" of Σ (§4). The skeleton matrix is calculated from the given description of Σ, and it underlies the statement of necessary and sufficient conditions for the existence of a corrective controller (§6).

An important aspect of our discussion is the creation of a mathematical framework for handling asynchronous machines with infinite cycles. This framework requires a generalization of the concept of state. Recall that, in qualitative terms, an asynchronous machine has two kinds of states: stable states, i.e., states in which the machine can linger indefinitely, and unstable states – transient states through which the machine passes in

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

International Journal of Control
ISSN 0020–7179 print/ISSN 1366–5820 online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207170600665022

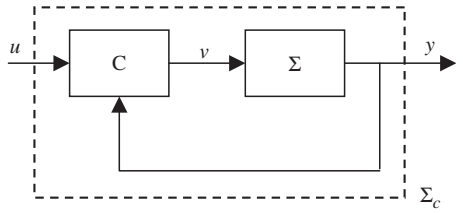


Figure 1. Basic control configuration.

quick succession. In § 4 we show that, in order to develop a consistent mathematical framework for asynchronous machines with infinite cycles, one must interpret an infinite cycle as a kind of "stable state", since a machine can linger indefinitely in an infinite cycle. This observation leads to the notion of a "generalized state", which is fundamental to the derivation of the skeleton matrix. It facilitates the statement of a simple necessary and sufficient condition for the existence of corrective controllers.

The discussion of this paper is a continuation of Hammer (1994, 1995, 1996a, b, 1997), Murphy *et al.* (2002, 2003), and Geng and Hammer (2004, 2005). The technical literature about infinite cycles of asynchronous machines deals mostly with techniques for the design and implementation of machines that are free of infinite cycles. These design techniques, initiated by Huffman (1954a, b, 1957), are reviewed in many textbooks on the design of digital systems, e.g., Kohavi (1970). It seems that the literature contains no reports regarding the use of control techniques to eliminate the effects of infinite cycles from an existing asynchronous sequential machine.

Much of the terminology and notation of the present paper follows Eilenberg (1974). Studies dealing with other aspects of the control of discrete event systems can be found in Ramadge and Wonham (1987), Dibenedetto *et al.* (1994), Thistle and Wonham (1994), Barrett and Lafortune (1998), Dibenedetto *et al.* (2001), the references cited in these papers, and others. Prior to the present paper and the ones preceding it in its sequence (Murphy *et al.* 2002, 2003, and Geng and Hammer 2004, 2005), there were apparently no reports addressing the impact of unstable states, critical races, or infinite cycles on the control of asynchronous machines. The earlier papers in the sequence deal with the development of controllers that eliminate the effects of critical races. The present paper concentrates on the design of controllers that eliminate the effects of infinite cycles, while guarantying that no critical races or other hazards are created when the control loop is closed.

The paper is organized as follows. Basic notation, terminology, and background are provided in § 2. An algorithm for characterizing the infinite cycles of a given asynchronous machine is described in § 3. Section 4 deals with the mathematical representation of machines with infinite cycles by introducing the notion of a generalized state. Our examination of corrective controllers for asynchronous machines with infinite cycles starts in § 5, where we discuss the skeleton matrix. Necessary and sufficient conditions for the existence of corrective controllers are presented in § 6, which also describes the construction of corrective controllers. The paper concludes with a comprehensive example (§ 7) and conclusions (§ 8).

2. Terminology and background

Let A be a finite non-empty alphabet, let A^* denote the set of all finite strings of characters of A, and let A^+ be the set of all non-empty strings in A^*. We assume that the alphabet A does not include the digits 0 and 1. The *length* $|w|$ of a string $w \in A^*$ is the number of characters of w. For two strings $w_1, w_2 \in A^*$, the *concatenation* is the string $w := w_2 w_1$, obtained by appending w_1 to the end of w_2 (note the reverse order). A *partial function* f: $S_1 \to S_2$ is a function whose domain is a subset of S_1, e.g., Eilenberg (1974).

An asynchronous machine Σ is defined by a sextuple (A, Y, X, x_0, f, h), where A, Y, and X are non-empty finite sets, x_0 is the initial state, and f: $X \times A \to X$ and h: $X \times A \to Y$ are partial functions. Here, A is the *input alphabet*, Y is the *output alphabet*, and X is the set of *states*. The partial function f is the *recursion function* and h is the *output function*. A *valid pair* $(x, u) \in X \times A$ is a point at which the partial functions f and h are defined.

The machine Σ starts from the initial state x_0 and accepts input strings of the form $u := u_0\, u_1 \cdots \in A^*$. In response, it generates a string of states $x_0 x_1 x_2 \cdots \in X^*$ and a string of output values $y_0 y_1 y_2 \cdots \in Y^*$, according to the recursion

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

An input sequence is *permissible* if all pairs (x_k, u_k), $k = 0, 1, 2, \ldots$ are valid pairs. The step counter k is incremented by one at every change of the input value or of the state value. The machine Σ is an *input/state machine* if $Y = X$ and the output is equal to the state at each step, i.e.,

$$y_k = x_k, \quad k = 0, 1, 2, \ldots.$$

An input/state machine Σ is represented by the triple (A, X, f), allowing for an arbitrary initial state.

A valid pair $(x, u) \in X \times A$ of the machine Σ is a *stable combination* if $f(x, u) = x$, i.e., if the state x is a fixed point of the function f. An asynchronous machine lingers at a stable combination until an input change occurs. A pair (x, v) that is not a stable combination is called a *transient combination*. A *potentially stable state* is a state for which there is a stable combination.

A transient pair (x, u) initiates a chain of transitions $x_1 = f(x, u)$, $x_2 = f(x_1, u), \ldots$, where the input character u is kept fixed. This chain of transitions may or may not end. If it ends, then there is an integer $q \geq 1$ such that the state $x' := f(x_q, u)$ of the chain satisfies $x' = f(x', u)$, i.e., (x', u) is a stable combination. In this case, the state x' is called the *next stable state* of x with the input value u. If this chain of transitions does not terminate, then the pair (x, u) is part of an *infinite cycle*.

The notion of next stable state leads to several important concepts, e.g., Kohavi (1970). The first of these is the *stable recursion function s*: $X \times A \to X$ of Σ, which is defined as follows. For every valid pair (x, u) of Σ that has a next stable state x', set $s(x, u) := x'$; leave s undefined for other pairs. Then, the *stable state machine* $\Sigma_{|s}$ induced by Σ is the sextuple (A, X, Y, x_0, s, h), where the stable recursion function s replaces the recursion function f of Σ. For transitions that do not involve infinite cycles, the stable state machine describes the behavior of Σ as observed by a user: it ignores all transients (which, ideally, occur in zero time) and highlights the persistent states of the machine (see also Murphy *et al.* 2003).

If the input value of an asynchronous machine changes while the machine is undergoing a chain of transitions, then the response of the machine may become unpredictable, since the state of the machine at the time of the input change is unpredictable. To avoid this uncertainty, asynchronous machines are normally operated in *fundamental mode*, where only one variable of the machine is allowed to change at a time; while one variable is undergoing transitions, all other variables of the machine are kept constant. In fundamental mode operation, a change of the input value is allowed only while the machine is in a stable combination. When the control configuration in figure 1 operates in fundamental mode, then the output of the controller C must remain constant while the machine Σ is undergoing state transitions, and the output of the machine Σ must remain constant while the controller C is undergoing state transitions.

Consider a machine Σ operating in fundamental mode and resting at a stable combination with the state x_1, when the input string $w = w_1 w_2 \ldots w_m$ is applied to it. The machine undergoes a chain of transitions, say, $x_2 = f(x_1, w_1)$, $x_3 = f(x_2, w_2), \ldots, x_{m+1} = f(x_m, w_m)$. Fundamental mode operation requires that

$$w_i = w_i + 1 \text{ whenever } (x_i, w_i) \text{ is not a stable combination, } i = 1, \ldots, m-1.$$

Thus, input values may be constant over a number of steps. Specifically, consider a situation where the input value of the machine Σ is kept constant at the character v, while the machine is engaged in a string of transitions through the states $x_1, x_2, x_3, \ldots$. Then, the machine goes through the state-input pairs (x_1, v), (x_2, v), $(x_3, v), \ldots$. In other words, the input value v is being repeated, as in $vvv\ldots$. To simplify notation, it is convenient to represent all such repetitions by one character v. In fact, this is how such input is implemented – the input value v is simply kept constant during the transition process. In this spirit, a string of the form $v_0 v_0 v_1 v_1 v_1 v_2 v_2$ is represented by the shortened form $v_0 v_1 v_2$, where each character is considered as being repeated once for each transition that occurs while it is in effect. Transitions with constant input are described by the following iteration.

For an integer $i \geq 1$ and a valid pair (x, u) of the machine Σ, we denote by $f^{\circ i}(x, u)$ the ith iteration of the recursion function f with the (constant) input character u, i.e.,

$$\begin{aligned} f^{\circ 1}(x, u) &:= f(x, u), \\ f^{\circ 2}(x, u) &:= f(f(x, u), u), \ldots, \\ f^{\circ i}(x, u) &:= f(f^{\circ i-1}(x, u), u), \quad i = 2, 3, \ldots. \end{aligned}$$

We refer to $f^{\circ i}$ as the ith constant input iteration of f.

Note that fundamental mode operation is impossible when a machine is in an infinite cycle, since the machine never reaches a stable combination with its active input value. To take a machine out of an infinite cycle, the input value of the machine must be changed during the cycle. As it is not possible to predict at which state the machine is when such an input change is applied, the outcome of an input change during an infinite cycle may be unpredictable. In our discussion, asynchronous machines operate in fundamental mode in all cases, except when in an infinite cycle. A detailed discussion of this point is provided in § 4 below.

We conclude our review with the following notion, which is critical to the discussion (see also Murphy *et al.* (2002, 2003) and Geng and Hammer (2005)).

Definition 1: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be two machines having the same input and the same output sets, and let $\Sigma_{|s}$ and $\Sigma'_{|s}$ be the stable state machines induced by Σ and Σ', respectively. Two states $x \in X$ and $\zeta \in X'$ are *stably equivalent* ($x \equiv \zeta$) if the following is true: when $\Sigma_{|s}$ starts from the state x

and $\Sigma'_{|s}$ starts from the state ζ, then (i) $\Sigma_{|s}$ and $\Sigma'_{|s}$ have the same permissible input strings; and (ii) $\Sigma_{|s}$ and $\Sigma'_{|s}$ generate the same output string for every permissible input string.

Two machines Σ and Σ' are *stably equivalent* if their initial states are stably equivalent, i.e., if $x_0 \equiv \zeta_0$; in such case, we write $\Sigma = \Sigma'$. □

Machines that are stably equivalent appear identical to a user.

3. Infinite cycles

As mentioned earlier, an infinite cycle occurs in an asynchronous machine when a valid pair (x, u) has no next stable state. In an infinite cycle, the machine keeps moving indefinitely from one transient combination to another, while the input character is kept constant. An infinite cycle can be characterized by listing the states and the input character involved in the cycle. Specifically, consider an asynchronous machine Σ with the state set $X = \{x^1, x^2, \ldots, x^n\}$. Assume that Σ has an infinite cycle χ that involves p states, say the states x^{j_1}, $x^{j_2}, \ldots, x^{j_p} \in X$ and the input character $a \in A$. The infinite cycle then functions according to the recursion

$$\begin{aligned} x^{j_{k+1}} &= f(x^{j_k}, a), \quad k = 1, \ldots, p-1, \\ x^{j_1} &= f(x^{j_p}, a). \end{aligned} \tag{1}$$

As a shorthand notation, we denote this infinite cycle by

$$\chi = \{x^{j_1}, x^{j_2}, \ldots, x^{j_p}; a\}.$$

The input character of χ is a, and the state set of χ is

$$X(\chi) := \{x^{j_1}, x^{j_2}, \ldots, x^{j_p}\}.$$

Using the iterated recursion function, we can write

$$X(\chi) = \{x^{j_1}, f(x^{j_1}, a), f^{\circ 2}(x^{j_1}, a), \ldots, f^{\circ(p-1)}(x^{j_1}, a)\}. \tag{2}$$

The *length* ℓ of the infinite cycle χ is the number of distinct states it includes, i.e., $\ell = p$ in this case.

Note that, when the length of an infinite cycle is 1, say $\chi = \{x; a\}$, only the second line of (1) applies, and it yields $x = f(x, a)$, i.e., (x, a) is stable combination of the machine Σ. Thus, an infinite cycle of length 1 is a stable combination. In the sequel, unless specifically stated otherwise, the term "infinite cycle" is used exclusively for infinite cycles of length greater than 1.

Of course, an asynchronous machine Σ can have more than one infinite cycle. The following statement shows that infinite cycles associated with the same input character must have disjoint state sets.

Lemma 1: *A valid state/input pair* (x, a) *of an asynchronous machine can be a member of at most one infinite cycle.*

Proof: Let χ_1 and χ_2 be two infinite cycles of the machine $\Sigma = (A, X, x_0, Y, f, h)$, and assume that both contain the pair (x, a). Let p be the length of χ_1, and let q be the length of χ_2. Then, using (2), the corresponding state sets are $X(\chi_1) = \{x, f(x, a), f^{\circ 2}(x, a), \ldots, f^{\circ(p-1)}(x,a)\}$ and $X(\chi_2) = \{x, f(x, a), f^{\circ 2}(x, a), \ldots, f^{\circ(q-1)}(x, a)\}$. Clearly, if $p = q$, then $X(\chi_1) = X(\chi_2)$, and $\chi_1 = \chi_2$. Otherwise, without loss of generality, assume that $p < q$. Applying (1) to the infinite cycle χ_1, it follows that $f^{\circ p}(x, a) = x$, so that $f^{\circ(p+1)}(x, a) = f(x, a)$, $f^{\circ p+2}(x, a) = f^{\circ 2}(x, a), \ldots, f^{\circ q}(x, a) = f^{\circ(q-p)}(x, a)$. Thus, $X(\chi_1) = X(\chi_2)$, and, since χ_1 and χ_2 also have the same input character a, we have $\chi_1 = \chi_2$. □

Lemma 1 allows us to derive a bound on the maximal number of infinite cycles an asynchronous machine can have, as follows. (Denote by $[q]^-$ the largest integer not exceeding q.)

Proposition 1: *Let* Σ *be an asynchronous machine with* n *states and an input alphabet of* m *characters. Then,* Σ *cannot have more than* $m[n/2]^-$ *infinite cycles.*

Proof: Consider the set of all infinite cycles involving a single input character a. In view of Lemma 1, all such infinite cycles have disjoint state sets; since an infinite cycle must contain at least 2 states, and since Σ has a total of n states, we conclude that Σ cannot have more than $[n/2]^-$ infinite cycles with the input character a. As this argument applies to each one of the m input characters of Σ, the total number of infinite cycles cannot exceed $m[n/2]^-$. □

In addition to infinite cycles, an asynchronous machine can have another malfeasance: critical races (Unger 1995). A critical race describes a situation where the next state of the machine is not uniquely determined. Specifically, a *critical race pair* (x, u) of an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ is a valid pair of Σ for which $f(x, u)$ is a subset of states, rather than a single state. In other words, for a machine with critical races, the recursion "function" f is multivalued at some points.

3.1 *Finding the infinite cycles of an asynchronous machine*

In order to control an asynchronous machine with infinite cycles, we must first determine all the infinite cycles of the machine from given data, i.e., from the recursion function of the machine. To this end, the following matrix is helpful.

Definition 2: Let $\Sigma = (A, X, x_0, Y, f, h)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$, and let ε be a character not included in A. Let U_{ij} be the set of input characters that take Σ in one step from the state x^j to the state x^i, i.e.,

$$U_{ij} = \{u \in A: x^i \in f(x^j, u)\}, \quad i, j = 1, \ldots, n.$$

Then, the *one-step transition matrix* $\tau(f)$ of Σ is an $n \times n$ matrix, whose (i, j) entry is

$$\tau_{ij}(f) := \begin{cases} U_{ij} & \text{if } U_{ij} \neq \varnothing, \\ \varepsilon & \text{if } U_{ij} = \varnothing, \quad i, j = 1, \ldots, n. \end{cases} \quad \square$$

Clearly, the information included in the one-step transition matrix is equivalent to the information included in the recursion function f of Σ. In fact, we can associate a one-step transition matrix $\tau(g)$ with every function g: $X \times A \to X$. The following fact is useful.

Lemma 2: *Let $\tau(f)$ be the one-step transition matrix of the asynchronous machine $\Sigma = (A, X, x_0, Y, f, h)$. If Σ has no critical races, then no input character appears more than once in a column of $\tau(f)$.*

Proof: By contradiction, assume that Σ has no critical races, while the input character u appears more than once in column j of the one-step transition matrix $\tau(f)$. Then, letting n be the number of states of Σ, there are two integers $p \neq q \in \{1, \ldots, n\}$ such that $u \in \tau_{pj}(f)$ and $u \in \tau_{qj}(f)$. Invoking the definition of $\tau(f)$, this means that $x^p \in f(x^j, u)$ and $x^q \in f(x^j, u)$. Consequently, (x^j, u) is a critical race pair of Σ, contradicting the fact that Σ has no critical races. This implies that column j of $\tau(f)$ cannot include more than one appearance of the input character u, and our proof concludes. $\square$

To simplify notation, we shall assume from now on that the machine Σ does not have any critical races. Still, critical races play an important role in our discussion, as they can be created when the machine is taken out of an infinite cycle. Furthermore, the results presented in this paper apply equally well to machines with critical races.

We define next some operations on the one-step transition matrix, which will allow us to examine the response of Σ over several steps. First, recalling that ε is not in A, we introduce the extended alphabet A', obtained by adding the character ε to A. Next, define the operations

$$\varepsilon \cup \varepsilon := \varepsilon,$$

$$u \cup \varepsilon := u \quad \text{for all } u \in A.$$

Note that ε behaves similarly to a "zero" under this operation. Further, define an operation of multiplication over the set A', given by

$$\begin{aligned} &u \circ u := u, \\ &u \circ u' := \varepsilon \quad \text{for all elements } u \neq u' \in A'. \end{aligned} \tag{3}$$

For subsets $\{a_1, a_2, \ldots, a_q\}$, $\{b_1, b_2, \ldots, b_r\} \subset A'$, the multiplication is defined pairwise over all possible pairs

$$\{a_1, a_2, \ldots, a_q\} \circ \{b_1, b_2, \ldots, b_r\} = \{a_i \circ b_j\}_{i=1,\ldots,q, j=1,\ldots,r} \tag{4}$$

Two operations on matrices whose entries are subsets of A' are needed. One is the union of two such $n \times n$ matrices C and D, defined entrywise by

$$(C \cup D)_{ij} := C_{ij} \cup D_{ij}, \quad i, j = 1, \ldots, n;$$

this operation is reminiscent of the numerical addition of matrices. The second operation is reminiscent of numerical multiplication of matrices

$$(C \circ D)_{ij} := \cup_{k=1,\ldots,n} C_{ik} \circ D_{kj} \quad \text{for all } i, j = 1, \ldots, n. \tag{5}$$

Example 1: For the matrices

$$C = \begin{pmatrix} a & \{a, b\} \\ \{b, c\} & b \end{pmatrix}; \quad D = \begin{pmatrix} b & c \\ a & a \end{pmatrix},$$

we obtain

$$C \circ D = \begin{pmatrix} a & a \\ b & c \end{pmatrix}. \quad \square$$

Constant input iteration is related to multiplication of the corresponding one-step transition matrices.

Lemma 3: *Let f: $X \times A \to X$ be a recursion function with the one-step transition matrix $\tau(f)$, and let $\tau(f^{\circ r})$ be the one-step transition matrix of the constant input iteration $f^{\circ r}$, $r = 2, 3, \ldots$. Then, $\tau(f^{\circ r}) = \tau(f^{\circ(r-1)}) \circ \tau(f)$.*

Before stating the proof, we provide an example.

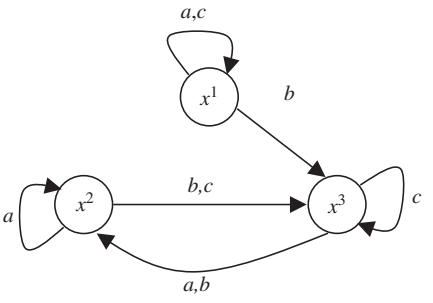
Example 2: Consider a machine Σ with the input set $A = \{a, b, c\}$, the state set $X = \{x^1, x^2, x^3\}$, and the transition function f. The state transition table and the state flow diagram of Σ are shown below in figure 2.

From the table, the one-step transition matrix of Σ is

$$\tau(f) = \begin{pmatrix} \{a, c\} & \varepsilon & \varepsilon \\ \varepsilon & a & \{a, b\} \\ b & \{b, c\} & c \end{pmatrix}. \tag{6}$$

	a	b	c
x^1	x^1	x^3	x^1
x^2	x^2	x^3	x^3
x^3	x^2	x^2	x^3

(*a*)



(*b*)

Figure 2. (a) State transition table of Σ; (b) State flow diagram of Σ.

A direct calculation yields

$$\tau(f^{\circ 2}) = \tau(f) \circ \tau(f) = \begin{pmatrix} \{a,c\} & \varepsilon & \varepsilon \\ b & \{a,b\} & a \\ \varepsilon & c & \{b,c\} \end{pmatrix} \quad (7)$$

□

Proof of Lemma 3: We start with $r = 2$. Assume that the state set X consists of the n elements $x^1, \ldots, x^n$. Using (5), we have

$$\tau(f) \circ \tau(f) = \cup_{k=1,\ldots,n} \tau_{ik}(f) \circ \tau_{kj}(f).$$

Clearly, each entry of $\tau(f) \circ \tau(f)$ is either ε, or it contains a character of the input alphabet A. Using (3), (4) and (5), it can be readily seen that the following two statements are equivalent for every character $u \in A$.

(i) The (i,j) entry of $\tau(f) \circ \tau(f)$ includes u.

(ii) There is an integer $k \in \{1, \ldots, n\}$ for which $u \in \tau_{ik}(f)$ and $u \in \tau_{kj}(f)$.

Now, when (ii) holds, we have $x^i = f(x^k, u)$ and $x^k = f(x^j, u)$, so that $x^i = f(f(x^j, u), u) = f \circ f(x^j, u) = f^{\circ 2}(x^j, u)$, and the (i,j) entry of $\tau(f^{\circ 2})$ includes u. Consequently, (ii) implies that

(iii) $u \in \tau_{ij}(f^{\circ 2})$.

Conversely, if (iii) holds, then it follows from the definition of $\tau(f^{\circ 2})$ that $x^i = f^{\circ 2}(x^j, u) = f(f(x^j, u), u)$. Now, the value $f(x^j, u)$ is an element of X, say the element x^k. Then, we have $x^k = f(x^j, u)$ and $x^i = f(x^k, u)$, which implies that (ii) is valid for this k. Thus, (iii) implies (ii), and, using our earlier observation, we conclude that (ii) is equivalent to (iii). As (i) and (ii) are also equivalent, it follows that (i) and (iii) are equivalent, and whence the matrices $\tau(f^{\circ 2})$ and $\tau(f) \circ \tau(f)$ have the same entries. This completes the proof for $r = 2$. The proof for $r > 2$ is done by induction, using a similar approach. □

Given an $n \times n$ matrix C whose entries are subsets of the alphabet A', introduce the powers

$$C^{\circ p} := C \circ C \circ \cdots \circ C \ (p \text{ times}), \quad p = 1, 2, \ldots$$

The following is a consequence of Lemma 3.

Corollary 1: *Let $\Sigma = (A, X, x_0, Y, f, h)$ be an asynchronous machine with the one-step transition matrix $\tau(f)$. Then, $\tau(f^{\circ p}) = \tau^{\circ p}(f)$ for every integer $p \geq 1$.* □

As discussed earlier, a stable combination (x, u) of an asynchronous machine Σ can be interpreted as an infinite cycle of length 1. We use this interpretation to simplify the wording of the following statement, which forms an important tool for characterizing all infinite cycles of an asynchronous machine.

Proposition 2: *Let Σ be an asynchronous machine with the recursion function f, the state set $X = \{x^1, \ldots, x^n\}$, and the input alphabet A. Then, the following three statements are equivalent for every valid pair $(x^i, u) \in X \times A$.*

(a) *There is an integer $q \geq 1$ for which u is included in the (i,i) entry of the matrix $\tau^{\circ q}(f)$.*

(b) *(x^i, u) is part of an infinite cycle, whose length ℓ is an integer divisor of q.*

(c) $x^i = f^{\circ q}(x^i, u)$.

Proof: First, to show that (a) implies (b), assume that u is included in the (i,i) entry of the matrix $\tau^{\circ q}(f)$. Considering that $\tau^{\circ q}(f) = \tau(f^{\circ q})$ by Corollary 1, it follows from the definition of the matrix $\tau(f^{\circ q})$ that

$$x^i = f^{\circ q}(x^i, u). \quad (8)$$

Hence, the set E of integers $p \geq 1$ for which $x^i = f^{\circ p}(x^i, u)$ is not an empty set. Let ℓ be the smallest non-zero integer in the set E. Then, $\ell \leq q$ by (8). Now, if $\ell = 1$, then $x^i = f(x^i, u)$, and (x^i, u) is a stable combination of Σ (i.e., an infinite cycle of length 1). If $\ell > 1$, then the equality $x^i = f^{\circ \ell}(x^i, u)$ implies that the infinite sequence of states x^i, $f(x^i, u)$, $f^{\circ 2}(x^i, u), \ldots$, constitutes a periodic sequence x^i, $f(x^i, u), \ldots, f^{\circ(\ell-1)}(x^i, u)$, x^i, $f(x^i, u), \ldots, f^{\circ(\ell-1)}(x^i, u), \ldots$, yielding an infinite cycle $\{x^i, f(x^i, u), \ldots, f^{\circ(\ell-1)}(x^i, u); u\}$ of length ℓ.

Now, using the integer division algorithm, we can write $q = a\ell + r$, where a and r are integers and $0 \leq r < \ell$. If $r > 0$, then the equality $x^i = f^{\circ \ell}(x^i, u)$ implies that

$$x^i = f^{\circ q}(x^i, u) = f^{\circ r}(f^{\circ a\ell}(x^i, u), u) = f^{\circ r}(x^i, u)$$

so that $x^i = f^{\circ r}(x^i, u)$. Considering that $r < \ell$, the last equality contradicts the fact that ℓ is the smallest nonzero integer in the set E. Thus, we must have $r = 0$, and ℓ is an integer divisor of q. This shows that (a) implies (b).

To show that (b) implies (c), assume that (b) is valid, i.e., that (x^i, u) is part of an infinite cycle whose length $\ell \geq 1$ is an integer divisor of q. Then, $q = a\ell$ for some integer $a \geq 1$, and the equality $x^i = f^{\circ\ell}(x^i, u)$ implies that $f^{\circ q}(x^i, u) = f^{\circ a\ell}(x^i, u) = [f^{\circ\ell}]^{\circ a}(x^i, u) = f^{\circ\ell}(f^{\circ\ell}(\cdots f^{\circ\ell}(x^i, u), \ldots, u), u) = x^i$, and (c) is valid.

Finally, we show that (c) implies (a). Assume that (c) is valid, so that $x^i = f^q(x^i, u)$ for some integer $q \geq 1$. Then, by Definition 2, the character u is included in the (i, i) entry of the matrix $\tau(f^q)$. As $\tau(f^q) = \tau^{\circ q}(f)$ by Corollary 1, it follows that u is included in the entry (i, i) of the power $\tau^{\circ q}(f)$ of $\tau(f)$, implying that (a) is valid. This concludes our proof. □

We describe now a procedure for finding all the states of an infinite cycle, when given one state of the cycle.

Proposition 3: *Let Σ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$, the recursion function f, and the one-step transition matrix $\tau(f)$. Assume that Σ has an infinite cycle χ of length $\ell > 1$ with the input character u, and let x^{j_0} be a state of χ. Then, the other states x^{j_1}, $x^{j_2}, \ldots, x^{j_{\ell-1}}$ of χ can be found by the following recursive process: having found the state x^{j_i}, the index j_{i+1} of the next state $x^{j_{i+1}}$ is the number of the row in which the character u appears in column j_i of $\tau(f)$, $i = 0, \ldots, \ell-2$.*

Proof: Noting that x^{j_0}, $x^{j_1}, \ldots, x^{j_{\ell-1}}$ are the states of our infinite cycle χ, it follows by (1) that $x^{j_{i+1}} = f(x^{j_i}, u)$, $i = 0, \ldots, \ell-2$, where u is the input character of χ. By Definition 2 of the one-step transition matrix $\tau(f)$, this means that the character u appears in position (j_{i+1}, j_i) of the matrix $\tau(f)$, which proves our statement. □

We will now outline an algorithm that finds all the infinite cycles of an asynchronous machine Σ. ($\#S$ denotes the number of elements of a set S.)

Algorithm 1: Let $\Sigma = (A, X, x_0, Y, f, h)$ be an asynchronous machine, and let $\tau(f)$ be its transition matrix. The steps below are performed individually for each input character $u \in A$.

Step 1: Let $\Delta_1(u)$ be the set of all states of Σ corresponding to diagonal entries of $\tau(f)$ that include the character u. Set $\delta_1(u) := \Delta_1(u)$.

Step 2: For $i \geq 2$, let $\Delta_i(u)$ be the set of all states of Σ corresponding to diagonal entries of $\tau^{\circ i}(f)$ that include the character u. Define the difference set

$$\delta_i(u) := \Delta_i(u) \backslash \bigcup\nolimits_{1 \leq j \leq i-1} \Delta_j(u). \tag{9}$$

Step 3: If

$$i + 1 > n - \sum_{j=1}^{i} \#\delta_j(u), \tag{10}$$

where n is the number of states of Σ, then the algorithm terminates for the character u. Otherwise, repeat from Step 2, using $i+1$ for i. □

As indicated in Theorem 1 below, the set $\delta_i(u)$ of Algorithm 1 consists of all states of the machine Σ that are included in infinite cycles of length i with the input character u. Note that, for a machine Σ with n states, it follows from (10) that Algorithm 1 cannot require more than n iterations for each input character. Here is an example.

Example 3: We demonstrate Algorithm 1 on the machine Σ of Example 2; the one-step transition matrix $\tau(f)$ of Σ is given by (6). Start by applying Algorithm 1 with the input character $a \in A$.

Step 1: Using (6) yields $\Delta_1(a) = \delta_1(a) = \{x^1, x^2\}$.

Step 2: Since $i < 2$, skip to Step 3.

Step 3: The inequality (10) becomes here $2 > 3 - 2 = 1$, which is true; hence, the Algorithm ends for the input character a.

We now proceed to the input character b.

Step 1: Since b does not occur on the main diagonal entry of $\tau(f)$, we have $\Delta_1(b) = \delta_1(b) = \varnothing$.

Step 2: Since $i < 2$, skip to Step 3.

Step 3: Inequality (10) becomes $2 > 3 - 0$, which is false; consequently, go to Step 2 with $i = 2$.

Returning to step 2: In view of (7), there are two occurrences of the input character b on the main diagonal entry of $\tau(f^{\circ 2})$: in positions (2, 2) and (3, 3). Consequently, $\Delta_2(b) = \{x^2, x^3\}$.

From (9),

$$\delta_2(b) = \Delta_2(b) \backslash \Delta_1(b) = \{x^2, x^3\}.$$

Returning to step 3: Inequality (10) becomes here $3 > 3 - 2 = 1$, which is true. Thus, Algorithm 1 terminates for the character b.
Finally, we proceed to input character c.

Step 1: From (6), we have $\Delta_1(c) = \delta_1(c) = \{x^1, x^3\}$.

Step 2: As $i = 1 < 2$, skip to Step 3.

Step 3: Inequality (10) becomes $2 > 3 - 2 = 1$, which is true; hence, Algorithm 1 terminates for the input character c. This terminates the entire process.

To summarize, we have obtained that $\delta_1(a) = \{x^1, x^2\}$, $\delta_1(b) = \varnothing$, $\delta_2(b) = \{x^2, x^3\}$, and $\delta_1(c) = \{x^1, x^3\}$. In view of Theorem 1 below, this implies that the states x^1 and x^2 form cycles of length 1 (i.e., stable combinations) with

the input character a; the states x^2 and x^3 form a cycle of length 2 with the input character b; and the states x^1 and x^3 form stable combinations with the input character c. □

Theorem 1: *Let $i \geq 1$ be an integer. Then, in the notation of Algorithm 1, the following are true.*

(a) *The set $\delta_i(u)$ consists of all states of the machine Σ that are members of infinite cycles of length i with the input character u.*
(b) *The machine Σ has exactly $\#\delta_i(u)/i$ infinite cycles of length i with the input character u.*

Proof: Let $X = \{x^1, \ldots, x^n\}$ be the state set of Σ, and let $u \in A$ be an input character. First, we show by induction that (9) can be rewritten in the form

$$\delta_i(u) := \Delta_i(u) \backslash \bigcup\nolimits_{1 \leq j \leq i-1} \delta_j(u), \quad i = 2, 3, \ldots \tag{11}$$

Indeed, since $\delta_1 = \Delta_1$ by Step 1 of Algorithm 1, the relation is valid for $i = 2$. Next, let $k \geq 2$ be an integer, and assume that (11) is valid for all $2 \leq i \leq k$. This implies that

$$\Delta_i = \bigcup\nolimits_{1 \leq j \leq i} \delta_j(u), \quad i = 2, 3, \ldots, k. \tag{12}$$

Now, considering $i = k + 1$ and substituting (12) into (9), we obtain that

$$\begin{aligned} \delta_{k+1}(u) &:= \Delta_{k+1}(u) \backslash \bigcup\nolimits_{1 \leq j \leq k} \Delta_j(u) \\ &= \Delta_{k+1}(u) \backslash \bigcup\nolimits_{1 \leq j \leq k} \delta_j(u). \end{aligned}$$

The last equality shows that (11) is valid for $i = k + 1$, and whence, by induction, it is valid for all $i \geq 2$.

Next, by Proposition 2, the set $\Delta_1(u)$ consists of all states of Σ that are included in infinite cycles of length 1 with the character u. As $\delta_1(u) = \Delta_1(u)$, this implies directly that part (a) of our present theorem is valid for $i = 1$. Preparing for induction, let $k \geq 1$ be an integer, and assume that part (a) of our theorem is valid for $i = k$. Proceeding to $i = k + 1$, it follows from (11) that

$$\delta_{k+1}(u) := \Delta_{k+1}(u) \backslash \bigcup\nolimits_{1 \leq j \leq k} \delta_j(u).$$

Now, by Proposition 2, the set $\Delta_{k+1}(u)$ consists of all states of Σ that are included in infinite cycles with the input character u, and whose length is an integer divisor of $k + 1$. Combining this with our induction assumption, it follows that the states included in $\delta_{k+1}(u)$ have the following properties:

(i) Being members of $\Delta_{k+1}(u)$, they are involved in infinite cycles with the input character u, and the length of these infinite cycles is an integer divisor of $k + 1$.
(ii) They are not involved in cycles whose length is k or less, since the states $\cup_{1 \leq j \leq k} \delta_j(u)$ are excluded from $\delta_{k+1}(u)$.

Thus, $\delta_{k+1}(u)$ consists exactly of all states that are included in infinite cycles of length $k + 1$ with the input character u, and part (a) of the theorem is valid.

Regarding Step 3 of Algorithm 1, we claim that the following is true. If the machine Σ has an infinite cycle χ' of length $i + 1$ with the input character u, then the states of this infinite cycle must be included in the difference set

$$D := X \backslash \bigcup\nolimits_{j=1,\ldots,i} \delta_i(u).$$

This is a consequence of the following facts: (i) by Lemma 1, the cycle χ' cannot have any common states with infinite cycles of length i or less that use the input character u; and (ii) by our earlier argument, $\cup_{j=1,\ldots,i} \delta_i(u)$ $\delta_i(u)$ includes the states of all infinite cycles of length i or less. But then, if $i + 1 > \#D$, the set D does not contain enough states to include an infinite cycle of length $i + 1$ or more, and whence the search for such infinite cycles terminates.

Turning to part (b) of the theorem, note that, by Lemma 1, different infinite cycles with the input character u have no states in common. Combining this with part (a), we conclude that the number of elements of $\delta_i(u)$ is a multiple of i, and the number of infinite cycles of length i is exactly $\#\delta_i(u)/i$, $i = 1, 2, \ldots$. □

The sets $\delta_i(u)$ derived in Algorithm 1 can be used to find all infinite cycles of the machine Σ by the following procedure.

Algorithm 2: In the notation of Algorithm 1, all infinite cycles of length i with the input character u are found as follows.

Step 0: If $\delta_i(u) = \varnothing$, then Σ has no infinite cycles of length i with the input character u, and the algorithm terminates. Otherwise, set

$$d_0 := \delta_i(u) \quad \text{and} \quad j := 0.$$

Step 1: Pick a state x^{j_0} from the set d_j. Using the procedure of Proposition 3, find the remaining $(i - 1)$ states $x^{j_1}, x^{j_2}, \ldots, x^{j_{i-1}}$ of the infinite cycle. Define the difference set

$$d_{j+1} := d_j \backslash \{x^{j_0}, x^{j_1}, x^{j_2}, \ldots, x^{j_{i-1}}\}.$$

Step 2: If $d_{j+1} = \varnothing$, then the algorithm terminates. Otherwise, repeat from Step 1, replacing j by $j + 1$. □

Algorithm 2 yields a complete characterization of all infinite cycles of an asynchronous machine Σ. It can be readily seen that, for a cycle of length i, the number

of iterations in Algorithm 2 cannot exceed $[n/i]^-$, where n is the number of states of the machine.

Example 4: We demonstrate Algorithm 2 on the machine Σ of Examples 2 and 3. Recall from Example 3 that $\delta_1(a)=\{x^1,\ x^2\}$, $\delta_1(b)=\varnothing$, $\delta_2(b)=\{x^2,\ x^3\}$, and $\delta_1(c)=\{x^1, x^3\}$.

Applying Algorithm 2 for the input character a:

Step 0: The relation $\delta_1(a)=\{x^1,\ x^2\}$ implies that the states x^1 and x^2 form stable combinations with the input character a. As $\delta_i(a)=\varnothing$ for $i>1$, this completes the consideration of this input character.

Next, apply Algorithm 2 to the input character b.

Step 0: $\delta_1(b)=\varnothing$, so there are no stable combinations with the input character b.

Step 0: $\delta_2(b)=\{x^2, x^3\}$, which means that there exist infinite cycles of length 2 with the input character b. Set

$$d_0 := \delta_2(b) \quad \text{and} \quad j := 0.$$

Step 1: Pick a state from d_0, say the state x^2. By Proposition 3, the state x^3 belongs to the same infinite cycle, yielding the infinite cycle $\{x^2, x^3; b\}$. As

$$d_1 := d_0 \backslash \{x^2, x^3\} = \varnothing,$$

the algorithm terminates for the input character b.

Finally, turning to the input character c, recall that $\delta_1(c)=\{x^1,\ x^3\}$ and $\delta_i(c)=\varnothing$ for all $i>1$. Thus, x^1 and x^3 form stable combinations with the input character c, and c is not involved in any (other) infinite cycles. □

4. Stable-state representations of machines with infinite cycles

Consider an asynchronous machine $\Sigma=(A, X, Y, x_0, f, h)$ that has no infinite cycles. For valid pairs (x, u) of Σ that have a next stable state x', we can define a partial function s: $X\times A\to X$ by setting $s(x, u) := x'$ for every valid pair (x, u) of Σ. The function s is called the stable recursion function of the machine Σ. When s is used as a recursion function, it induces the stable-state machine $\Sigma_{|s}=(A, X, Y, x_0, s, h)$. The stable-state machine describes the persistent states of Σ, and hence describes the behavior of Σ as experienced by a user (see also Murphy *et al.* (2002, 2003)).

For an asynchronous machine Σ with infinite cycles, being in an infinite cycle is clearly a persistent status of the machine, and this status is definitely experienced by the machine's user. Thus, if the notion of stable-state machine is to remain true to its goal of representing the persistent features of Σ, then it must include a representation of the infinite cycles of Σ. This leads to the following generalization of the notion of a stable-state machine, which is critical to the development of control strategies.

Definition 3: Let Σ be an asynchronous machine with the state set $X=\{x^1, \ldots, x^n\}$. Assume that Σ has $t>0$ infinite cycles $\chi_1, \ldots, \chi_t$ of length greater than 1. With each infinite cycle χ_i of Σ, associate a new state x^{n+i}, called a cycle state. The set

$$\tilde{X} := \{x_i, \ldots, x^n, x^{n+1}, \ldots, x^{n+i}\}$$

is called the *augmented state set* of Σ. The elements of are the *generalized states* of Σ.

A pair (x, u) $\tilde{X}\times A$ is a *generalized valid pair* of Σ if one of the following holds: (i) $x\in X$ and (x, u) is a valid pair of Σ; or (ii) $x=x^{n+i}$ for an integer $i\in\{1, \ldots, t\}$ and u forms a valid pair with each state of the infinite cycle χ_i.

A pair $(x, u)\in\tilde{X}\times A$ is a *generalized stable combination* of Σ if one of the following is valid: (i) $x\in X$ and (x, u) is a stable combination of Σ; or (ii) $x=x^{n+i}$ for an integer $i\in\{1, \ldots, t\}$ and u is the input character of the infinite cycle χ_i. □

Thus, any persistent status of the machine Σ is described by a generalized stable combination, since a persistent status of Σ is either a stable combination or an infinite cycle.

Example 5: Consider the machine Σ of Example 4, where it was shown that Σ has only one infinite cycle (of length bigger than one) – the infinite cycle $\chi^1 := \{x^2, x^3; b\}$. With this infinite cycle, we associate a cycle state x^4. The augmented state set of Σ is then $\tilde{X}=\{x^1, x^2, x^3, x^4\}$. The generalized valid pairs of Σ include, for instance, the pairs (x^1, a), (x^2, a), (x^3, a), and (x^4, a). The generalized stable combinations of Σ are (x^4, b), (x^1, a), (x^2, a), (x^1, c), and (x^3, c). □

We introduce now a new recursion function over the augmented state set of the asynchronous machine $\Sigma=(A, Y, X, x_0, f, h)$. Let $\chi_1, \ldots, \chi_t$ be the infinite cycles of Σ, and let $\tilde{X}=\{x^1, \ldots, x^{n+t}\}$ be its generalized state set. First, we define a partial function s^e: $X\times A\to\tilde{X}$ over all valid pairs $(x, u)\in X\times A$ of Σ by setting

$$s^e(x,u)=\begin{cases} s(X,u) & \text{if } (x,u) \text{ has a next stable state,}\\ x^{n+i} & \text{if } (f(x,u),u) \text{ is a pair of the}\\ & \quad \text{infinite cycle } \chi_i.\end{cases}$$

Recall that, for an infinite cycle χ of Σ, the symbol $X(\chi)$ indicates the set of states included in χ. Let u be an input character that forms valid pairs with all states of χ. Denote by $s^e[X(\chi), u]$ the image of the set $X(\chi) \times u$ through s^e, namely,

$$s^e[X(\chi), u] := \{x' \in \tilde{X} : x' = s^e(x, u) \quad \text{and} \quad x \in X(\chi)\}.$$

Note that $s^e[X(\chi), u]$ can be a single state or a set of states, depending on χ and on u. The following notion is critical to the control of asynchronous machines with infinite cycles. It defines a function whose values are subsets of $\tilde{X}$. When the value consists of a single element x of $\tilde{X}$, we shall drop the (formal) distinction between the element $x \in \tilde{X}$ and the subset $\{x\} \subset \tilde{X}$.

Definition 4: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the infinite cycles $\chi_1, \ldots, \chi_t$ and the augmented state set $\tilde{X} = \{x^1, \ldots, x^{n+t}\}$. Denote by $P(\tilde{X})$ the set of all subsets of $\tilde{X}$. Then, the *generalized stable recursion function* s: $\tilde{X} \times A \to P(\tilde{X})$ of Σ is defined over all generalized valid pairs $(x, u) \in \tilde{X} \times A$ of Σ by

$$s(x, u) := \begin{cases} s^e(x, u) & \text{if } x \in X, \\ s^e[X(\chi_i), u] & \text{if } x = x^{n+i},\ i = 1, \ldots, t. \end{cases}$$

The *generalized stable-state machine* $\Sigma_{|s} = (A, \tilde{X}, s)$ of Σ is an input/state machine with the state set $\tilde{X}$ and the recursion function s. □

Note that, when the machine Σ has no infinite cycles, the generalized stable recursion function is essentially equal to the stable recursion function (only the codomains differ). In such case, we will make no distinction between the two functions. However, when Σ has infinite cycles, the generalized stable recursion function can be "multivalued" over cycle states.

Example 6: Referring to the machine Σ of Example 5, the generalized stable state machine $\Sigma_{|s} = (A, \tilde{X}, s)$ is given in table 1. The state x^4 represents an infinite cycle. □

The generalized stable transition function describes the behavior of an asynchronous machine as experienced by a user, since it describes the persistent aspects of the machine's response. We are now ready to state in formal terms the main problem considered in this paper. Denote by $\Sigma_{c|s}$ the generalized stable-state machine induced by the closed loop system Σ_c of figure 1.

The model matching problem: Let Σ and Σ' be input/state asynchronous machines having the same input and output alphabets, where Σ' is a stable-state machine. Find necessary and sufficient conditions for the existence of a controller C for which the stable-state machine $\Sigma_{c|s}$ is stably equivalent to Σ' for all initial conditions. When C exists, provide a method for its design. □

Table 1. Generalized stable transition table of Σ.

	a	b	c
x^1	x^1	x^4	x^1
x^2	x^2	x^4	x^3
x^3	x^2	x^4	x^3
x^4	x^2	x^4	x^3

The controller C of the model matching problem makes the closed loop system simulate the stable-state behavior of the machine Σ'. The machine Σ' is called the *model*, and it has no infinite cycles. Thus, when model matching is achieved, the controller C eliminates the effects of the infinite cycles of Σ. Model matching for asynchronous machines without infinite cycles was discussed in Murphy *et al.* (2002, 2003) and Geng and Hammer (2004, 2005). The presence of infinite cycles requires the development of new analytical tools, including the notions of generalized state and generalized stable recursion function introduced earlier in this section. We assume that Σ and Σ' have the same state set and the same initial condition.

4.1 *The generalized skeleton matrix*

For asynchronous machines with infinite cycles, fundamental mode operation is usually impossible, since exiting an infinite cycle requires an input change while the cycle is in progress, i.e., while the system is in transition. For such machines, the closest one can come to fundamental mode operation is described by the following notion.

Definition 5: An asynchronous machine Σ operates in *semi-fundamental mode* if it operates in fundamental mode when not in an infinite cycle. □

Recall that, in the generalized stable state machine $\Sigma_{|s}$ of Σ, an infinite cycle χ of Σ is represented by a stable combination (x, u), where x is the cycle state corresponding to χ and u is the input value of χ. Accordingly, the following is valid.

Proposition 4: *Semi-fundamental mode operation of an asynchronous machine* Σ *is equivalent to fundamental mode operation of the generalized stable state machine* $\Sigma_{|s}$ *induced by* Σ. □

We adjust now the notion of stable reachability (Murphy 2002, 2003) to our present framework.

Definition 6: Let Σ be an asynchronous machine with the augmented state set $\tilde{X}$ and the generalized stable recursion function s. A generalized state $x' \in \tilde{X}$ is *stably reachable* from a generalized state $x \in \tilde{X}$ if there

is a input string $u = u_0 u_1 \ldots u_k$ of Σ for which $x' \in s(x, u)$. □

To perform computations related to stable reachability, we need the following matrix.

Definition 7: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and the infinite cycles $\chi_1, \ldots, \chi_t$. Let $\Sigma_{|s} = (A, \tilde{X}, s)$ be the generalized stable state machine induced by Σ, where $\tilde{X} = \{x^1, \ldots, x^{n+t}\}$ and x^{n+i} is the generalized state corresponding to the infinite cycle χ_i, $i = 1, \ldots, t$. Denote by $s^*(x^i, x^j)$ the set of all input characters $u \in A$ for which $x^i \in s(x^j, u)$, and let N be a character not included in the alphabet A. Then, the *matrix of one-step generalized stable transitions* $R(\Sigma_{|s})$ is an $(n+t) \times (n+t)$ matrix whose (i,j) entry is given by

$$R_{ij}(\Sigma_{|s}) = \begin{cases} s^*(x^i, x^j) & \text{if } s^*(x^i, x^j) \neq \varnothing, \\ N & \text{otherwise,} \end{cases}$$

$i, j = 1, \ldots, n+t$. □

The matrix $R(\Sigma_{|s})$ characterizes the set of all one step transitions of the generalized stable-state machine $\Sigma_{|s}$. Its (i,j) entry, if not N, consists of all (single) input characters that can take $\Sigma_{|s}$ from x^j to a generalized stable combination with x^i. An (i,j) entry of N indicates that $\Sigma_{|s}$ cannot be driven from x^j to a generalized stable combination with x^i by applying a single input character.

Example 7: A direct calculation shows that the matrix of one-step generalized stable transitions of the machine considered in Example 6 is given by

$$R(\Sigma_{|s}) = \begin{pmatrix} \{a, c\} & N & N & N \\ N & a & a & a \\ N & c & c & c \\ b & b & b & b \end{pmatrix}. \tag{13}$$

□

We describe now some operations on the matrix $R(\Sigma_{|s})$ that are essential to the solution of the model matching problem for asynchronous machines, starting with an operation that mimics matrix addition (Murphy *et al.* (2002, 2003). Let A^* be the set of all strings of characters of the alphabet A, and let w_i be a subset of A^* or the character N, $i = 1, 2$. The operation Ψ of *unison* is defined by ($\cup$ indicates union of sets)

$$w_1 \Psi w_2 := \begin{cases} w_1 \bigcup w_2 & \text{if } w_1 \subset A^* \text{ and } w_2 \subset A^*, \\ w_1 & \text{if } w_1 \subset A^* \text{ and } w_2 = N, \\ w_2 & \text{if } w_1 = N \text{ and } w_2 \subset A^*, \\ N & \text{if } w_1 = w_2 = N. \end{cases}$$

Note that N is treated like the empty set it represents. The unison $C := A \Psi B$ of two $n \times n$ matrices A and B, whose entries are either subsets of A^* or the character N, is defined entrywise by $C_{ij} := A_{ij} \Psi B_{ij}$, $i, j = 1, \ldots, n$.

Concatenation of elements $w_1, w_2 \in A^* \cup N$ is defined by

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

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

$$\text{conc}(W, V) := \Psi_{\substack{i=1,\ldots,q \\ j=1,\ldots,r}} \text{conc}(w_i, v_j).$$

Note that the concatenation result is either a subset of A^* or the character N. The concatenation is non-commutative, and N takes the role of a "zero".

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

$$Z_{ij} := \Psi_{k=1}^{n} \text{conc}(C_{ik}, D_{kj}), \; i, j = 1, \ldots, n.$$

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

$$R^q(\Sigma_{|s}) := R^{q-1}(\Sigma_{|s}) R(\Sigma_{|s}), \; q = 2, 3, \ldots.$$

For an integer $q \geq 1$, the matrix $R^q(\Sigma_{|s})$ has the following physical significance. If not N, the (i,j) entry of $R^q(\Sigma_{|s})$ is the set of all input strings that may take x^j to a generalized stable combination with x^i in exactly q generalized stable transitions (these transitions can result in a critical race). If the (i,j) entry is N, then it is not possible to reach from x^j to a generalized stable combination with x^i in exactly q generalized stable transitions. Thus, $R^q(\Sigma_{|s})$ is called the *matrix of q-step generalized stable transitions*. Define

$$R^{(q)}(\Sigma_{|s}) := \Psi_{p=1\ldots q} R^p(\Sigma_{|s}) \; q = 2, 3, \ldots. \tag{14}$$

By construction, if not N, the (i,j) entry of $R^{(q)}(\Sigma_{|s})$ consists of all strings that may take the machine $\Sigma_{|s}$ from x^j to a generalized stable combination with x^i in q or fewer generalized stable transitions (these transitions can result in a critical race). The following fact is important to our discussion; its proof is similar to that of Murphy *et al.* (2003, Lemma 3.9).

Lemma 4: *Let Σ be an asynchronous machine with n states and t infinite cycles, and let $\Sigma_{|s}$ be the generalized stable-state machine induced by Σ. Then, the following two statements are equivalent.*

(a) *The generalized state x^i is stably reachable from the generalized state x^j.*
(b) *The (i,j) entry of $R^{(n+t-1)}(\Sigma_{|s})$ is not N.* □

Thus, the matrix $R^{(n+t-1)}(\Sigma_{|s})$ characterizes all transitions possible for the generalized stable-state machine $\Sigma_{|s}$ induced by Σ. These include transitions that start at infinite cycles and transitions that pass through infinite cycles. The latter two classes of transitions may culminate in critical races. Transitions that culminate in a critical race can be easily recognized in the matrix $R^{(n+t-1)}(\Sigma_{|s})$, as discussed below. First, we introduce Definition 8.

Definition 8: Let $R(\Sigma_{|s})$ be the matrix of one-step generalized stable transitions of the machine Σ, where Σ has n states and t infinite cycles. The *generalized stable reachability matrix* of Σ is $\Gamma(\Sigma) := R^{(n+t-1)}(\Sigma_{|s})$. □

When the machine Σ is in an infinite cycle χ^i, it moves quickly from state to state within the cycle. A change of the input character during the infinite cycle may lead to an unpredictable outcome, since the exact state at which the input change occurs is unpredictable. When translated into our generalized framework, this means that a cycle state, such as the cycle state x^{n+i} corresponding to χ^i, may be involved in critical races.

For example, assume that $\chi^i = \{x^1, x^2; u\}$, and that the input character is changed from u to v, where $s(x^1, v) = x^3$, while $s(x^2, v) = x^4$. As Σ switches quickly between the states x^1 and x^2 during the infinite cycle χ^i, it is not possible to predict in which one of these states Σ will be when the input character changes to v. Thus, it is impossible to foretell whether the outcome of the change to v will be x^3 or x^4; this indicates that the pair (x^{n+i}, v) is a critical race pair.

In the one-step matrix of generalized stable transitions $R(\Sigma_{|s})$, a critical race is represented by the presence of the same input character in different entries of a column. For instance, in our example, the input character v will appear in rows 3 and 4 of column $n+i$ of $R(\Sigma_{|s})$. Combining these observations with Lemma 4, we reach the following conclusion.

Proposition 5: *Let Σ be an asynchronous machine with n states and t infinite cycles, and let $\Gamma(\Sigma)$ be its generalized stable reachability matrix. Then, the following are equivalent for all input strings $u \in A^+$ and for all $j = 1, \ldots, n+t$.*

(i) *Applying u at the generalized state x^j results in a critical race.*
(ii) *The string u appears in more than one entry of column j of the matrix $\Gamma(\Sigma)$.* □

Definition 9: Let $\Sigma_{|s}$ be a generalized stable state machine with the generalized stable recursion function s, and let u be an input string of $\Sigma_{|s}$. The transition induced by u from a generalized state x is a *deterministic transition* if $s(x,u)$ consists of a single state. The machine Σ is a *deterministic machine* if all transitions of $\Sigma_{|s}$ are deterministic. □

Consider again the machine Σ with the generalized state set $\{x^1, \ldots, x^{n+t}\}$ and the input alphabet A. Assume that the input string $u \in A^+$, when applied at the generalized state x^j of $\Sigma_{|s}$, creates a critical race with the outcomes x^p and x^q, where $p \neq q$. Assume further that there are input strings that take the machine Σ from these two states to a common target state x^s via deterministic transitions. Namely, assume there are input strings u^1, $u^2 \in A^+$ such that u^1 takes $\Sigma_{|s}$ from x^p to x^s deterministically, while u^2 takes $\Sigma_{|s}$ from x^q to x^s deterministically; i.e., $x^s = s(x^p, u^1) = s(x^q, u^2)$. Then, by using state-feedback control, we can induce a deterministic transition from x^j to x^s as follows: apply the input string u at the state x^j, and check the outcome. If the outcome is x^p, then continue with the input string u^1; if the outcome is x^q, then continue with the input string u^2. This strategy leads to the deterministic outcome x^s, masking the effects of the critical race along the way; it creates a deterministic transition from x^j to x^s. More generally, we have the following concept. (Denote by Π_x: $\tilde{X} \times A \to \tilde{X}$: $\Pi_x(x,u) = x$ the standard projection onto the state set.)

Definition 10: Let Σ be an asynchronous machine with the generalized state set $\tilde{X} = \{x^1, \ldots, x^{n+t}\}$, the input alphabet A, and the generalized stable recursion function s. A *feedback trajectory* from the generalized state x^j to the generalized state x^i is a list $\{S_0, S_1, S_2, \ldots, S_p\}$ of sets of valid pairs of $\Sigma_{|s}$ with the following properties:

(i) $S_0 = \{(x^j, u_0)\}$,
(ii) $s[S_\alpha] \subset \Pi_x[S_{\alpha+1}]$, $\alpha = 0, \ldots, p-1$,
(iii) $s[S_p] = \{x^i\}$. □

Example 8: Consider the generalized stable state machine $\Sigma_{|s}$ of Example 6. By iterating the generalized stable recursion function induced by Table 1, we obtain the following feedback trajectory from the generalized

state x^1 to the generalized state x^2: $S_0 = \{(x^1, b)\}$, $S_1 = \{(x^4, a)\}$, $S_2 = \{(x^2, a)\}$. □

A feedback trajectory characterizes the existence of a feedback controller, as follows.

Proposition 6: *Let Σ be an asynchronous machine, and let x^j and x^i be two generalized states of Σ. The following two statements are equivalent.*

(a) *There is a state feedback controller C that drives Σ through a deterministic transition from x^j to x^i, using semi-fundamental mode operation.*
(b) *There is a feedback trajectory from x^j to x^i.*

Proof: Assume first that (b) is valid, and let $\{S_0, S_1, S_2, \ldots, S_p\}$ be a feedback trajectory from x^j to x^i. We build now a state feedback controller that takes the machine Σ deterministically from x^j to x^i in response to a command input character. As depicted in figure 1, the controller has two inputs: one is the state of the controlled machine Σ, and the other is an external input u that serves as the command input of the controller. Given a set of characters $W \subset A$, we build a controller $C(x^j, x^i, W)$. It drives the machine Σ from the state x^j to a generalized stable combination with the state x^i, in response to an input character $w \in W$. Assume then that Σ is in a generalized stable combination with the state x^j. Referring to figure 1, the controller $C(x^j, x^i, W)$ rests in its initial state as long as $u \notin W$, feeding the input character u to Σ (i.e., C is then transparent). When u changes into one of the characters of W, then C starts to feed into the input of Σ a string of characters that take Σ from x^j to x^i through a string of generalized stable transitions, i.e., through a string of transitions of $\Sigma_{|s}$. In view of Proposition 4, stable transitions of $\Sigma_{|s}$ imply semi-fundamental mode operation of Σ.

Denote by ϕ the recursion function of $C(x^j, x^i, W)$ and by η its output function, and let ξ denote a state of $C(x^j, x^i, W)$. Recalling that $C(x^j, x^i, W)$ has two inputs – the generalized state x of Σ and the external input u – it follows the functions ϕ and η depend on the three variables ξ, x, and u. To indicate this fact, we shall write $\phi(\xi,(x,u))$ and $\eta\ (\xi,(x,u))$.

Let $U(x^j) \subset A$ be the set of all input characters that form stable combinations with the generalized state x^j, and let ξ_0 designate the initial state of $C(x^j, x^i, W)$. We design $C(x^j, x^i, W)$ to move to the state $\xi_1(x^j)$ when it detects a stable combination with the generalized state x^j, in preparation for a possible activation. To implement, set

$$\phi(\xi_0, (z,t)) := \xi_0 \quad \text{for all } (z,t) \in X \times A \backslash x^j \times U(x^j),$$
$$\phi(\xi_0, (x^j, u)) := \xi_1(x^j) \text{ for all } u \in U(x^j).$$

When in the state ξ_0, the controller $C(x^j, x^i, W)$ is transparent – it applies its own external input u to Σ, so we set

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

We refer to the state $\xi_1(x^j)$ as the *transition state* of the controller C. At the transition state $\xi_1(x^j)$, the controller feeds into Σ an input character that forms a stable combination with x^j, so that Σ continues to rest at x^j until further change. To implement, choose a character $u_j \in U(x^j)$, and set

$$\eta(\xi_1(x^j), (x^j, t)) := u_j \quad \text{for all } t \in U(x^j).$$

The purpose of the transition state is to decouple the machine Σ from the external input of figure 1 when it reaches a stable combination with the state x^j. This shields the machine when the external input changes to a character of W, allowing the controller to take over the machine's input without disturbing semi-fundamental mode operation.

Assume further that Σ is in a stable combination with the generalized state x^j, when a command input character $w \in W$ is applied to the controller. This initiates the process of taking Σ from x^j to x^i. Our forthcoming construction uses P new controller states to implement the transition from x^j to x^i, where

$$P = \#\Pi_x S_0 + \#\Pi_x S_1 + \cdots + \#\Pi_x S_p. \tag{15}$$

Each one of the new controller states is denoted by $\xi^k(x^j, w, x)$, where $x \in \Pi_x S_k$ and $k = 1, \ldots, p$. As Σ is in a generalized stable combination with x^j, the controller is at the state $\xi_1(x^j)$. When the input character w appears, the controller transits to the state $\xi^0(x^j, w, x^j)$; this is accomplished by setting

$$\phi(\xi_1(x^j), x^j, w) := \xi^0(x^j, w, x^j) \quad \text{for all } w \in W$$
$$\phi(\xi_1(x^j), x^j, u) := \xi_1(x^j) \quad \text{for all } u \in U(x^j) \backslash W;$$
$$\phi(\xi_1(x^j), x^j, u) := \xi_0 \quad \text{for all } u \notin U(x^j) \cup W.$$

Let $u_0 \in A$ be an input character for which $(x^j, u_0) \in S_0$. Set the controller output to

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

i.e., the controller $C(x^j, x^i, W)$ applies u_0 to Σ. By the definition of our feedback trajectory, this makes Σ

move to a generalized stable combination with a generalized state in the set $\Pi_x S_1$, say to the generalized state x_1. Let $u_1 \in A$ be an input character for which $(x_1, u_1) \in S_1$. When Σ reaches x_1, the controller moves to its next state $\xi^1(x^j, w, x_1)$; this is implemented by setting:

$$\phi(\xi^0(x^j, w, x^j), x_1, w) := \xi^1(x^j, w, x_1).$$

The controller output function is defined by

$$\eta(\xi_1(x^j, w, x_1), x, v) := u_1 \quad \text{for all } (x, v) \in X \times A.$$

This guarantees semi-fundamental mode operation, since the controller output changes only after Σ has reached a generalized stable combination with x_1. When applied to the machine Σ, the input value u_1 takes Σ to a generalized stable combination with a generalized state in the set $\Pi_x S_2$, say to the generalized state x_2. Let $u_2 \in A$ be an input character for which $(x_2, u_2) \in S_2$. The process continues similarly until x^i is reached. In general, at a step $k \in \{1, 2, \ldots, p\}$, the machine Σ reaches the generalized state $x_k \in \Pi_x S_k$, and there is an input character $u_k \in A$ for which $(x_k, u_k) \in S_k$. Following the above pattern, the transition function and the output function of $C(x^j, x^i, W)$ are defined by (to simplify notation, set $x_0 := x^j$)

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

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

Note that this definition assures semi-fundamental mode operation.

At $k = p$, the machine Σ reaches a state $x_p \in \Pi_x S_p$. Let u_p be an input character such that $(x_p, u_p) \in S_p$. According to the definition of the feedback trajectory, we have $s(x_p, u_p) = x^i$. Consequently, set

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

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

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

In this way, the controller keeps Σ at the stable combination (x^i, u_p) as long as the external input character remains within W. When the external input character leaves W, the controller resets to its initial condition ξ_0. This completes the construction of the controller and shows that (b) implies (a). Note that the total number of controller states is $P + 2$, where P is given by (15), and the additional two states are ξ_0 and $\xi_1(x^j)$.

Conversely, assume that (a) is valid. Let Ξ be the state set of the controller C, and let ξ_0 be its initial state. Denote by $C(\xi, x, u)$ the output value produced by the controller C when it is at the next stable state corresponding to its state ξ, the generalized state x of Σ, and the external input value u. By assumption, there is an external input value w that induces the controller C to generate an input string $u_0 u_1 \ldots u_p$ for Σ, taking Σ from the generalized state x^j to the generalized state x^i in semi-fundamental mode operation, with deterministic outcome. The first character of this input string is $u_0 = C(\xi_0, x^j, w)$. Define the set $S_0 := \{(x^j, u_0)\}$.

Let s be the generalized stable recursion function of Σ. When Σ receives the input value u_0 from the controller, it moves to a generalized stable combination with one of the states of the set $s(x^j, u_0) = s[S_0]$. When Σ reaches this state, the controller C, which operates in semi-fundamental mode, detects the new state of Σ and moves to its own next stable state. Let $\xi(x, u_0)$ denote the next stable state reached by C immediately following a transition of Σ to a generalized stable combination with a state $x \in s(x^j, u_0)$. Let $u_1 = C(\xi(x, u_0), x, w) \in A$ be the output character generated by the controller upon reaching $\xi(x, u_0)$. Define the set

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

Continuing in this way, assume that we have defined the set S_k for an integer $k \geq 0$. We can then build a new set by setting

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

By assumption, the controller C drives Σ (deterministically) to the state x^i. Consequently, there is an integer p such that $s(S_p) = x^i$. In view of our construction, the list $S_0, S_1, \ldots, S_p$ forms a feedback trajectory. Thus, the existence of a state feedback controller C that drives Σ through a deterministic transition from x^j to x^i in semi-fundamental mode operation, implies the existence of a feedback trajectory from x^j to x^i. This shows that (a) implies (b), and our proof concludes. □

The proof of Proposition 6 contains an algorithm for the construction of a state feedback controller that takes the machine Σ between two states among which there is a feedback trajectory. After constructing the controller according to the algorithm, the number of states can often be reduced by using standard machine reduction techniques.

We turn now to the issue of characterizing the set of all pairs of generalized states of Σ that can be connected by a feedback trajectory. The next algorithm characterizes all such pairs of generalized states. The algorithm consists of a string of operations on the matrix of one-step generalized stable transitions. It gradually transforms the matrix into a numerical matrix with entries of zero and one, called the "generalized skeleton matrix" of Σ. We shall see later that, in this matrix, an entry of 1 appears in position (i,j) if and only if there exists a state-feedback controller that induces a deterministic transition from x^j to x^i. The result is a simple and concise characterization of the ways in which a state feedback controller can affect an asynchronous machine Σ.

Before continuing, we define the following *meet* operation involving strings of A^+ and the digits 0 and 1. Let ω be a character not included in A. Set

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

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

for all $a \in \mathrm{A}^+$.

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

$$\begin{pmatrix} 1 \\ 1 \\ 0 \\ a \\ 0 \end{pmatrix} \bigwedge \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ a \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \omega \\ 0 \end{pmatrix}.$$

Algorithm 3: Let $\Gamma(\Sigma)$ be the generalized stable reachability matrix of the asynchronous machine 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 $i,\ j = 1, \ldots, n+t$; then continue to (b):

(a) If K^1_{ij} includes a string of A^+ that does not appear in any other entry of the same column j, then perform the following operations:

Delete any string included in K^1_{ij} from all entries of column j 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 $k := 1$, $\alpha := 1$.

Step 3: If $k = n+t+1$, then perform the following operations.

(a) Set $K_\alpha(\Sigma) := K(k)$.
(b) If $\alpha \geq 2$ and $K_\alpha(\Sigma) = K_{\alpha-1}(\Sigma)$, then replace by 0 all entries of $K_\alpha(\Sigma)$ that are not 1; denote the resulting matrix by $K_g(\Sigma)$, and terminate the Algorithm. Otherwise, replace α by $\alpha+1$, set $k := 1$, and continue to Step 4.

Step 4: If all entries of column 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:

(a) If there is a character $u \in A$ that appears in column k of $K(k)$, then let $i_1, i_2, \ldots, i_q$ be the rows of column k of $K(k)$ that include u. Denote by $J(u)$ the meet of columns $i_1, i_2, \ldots, i_q$ of the matrix $K(k)$.
(b) If $J(u)$ has no entries other than 0 or 1, then delete u from all entries of column k of the matrix $K(k)$; set all empty entries, if any, to the value 0. Continue to (c).
(c) If $J(u)$ has no entries of 1, then return to Step 4. Otherwise, continue to (d).
(d) If $J(u)$ has entries of 1, then let $j_1, \ldots, j_r$ be the entries of $J(u)$ having the value 1. Let $S(k)$ be the set of columns of $K(k)$ that consists of column k and of every column that has the number 1 in row k. In the matrix $K(k)$, perform the following operations on every column of $S(k)$.
(1) Delete from the column all occurrences of input characters that appear in rows $j_1, \ldots, j_r$ of the column.
(2) Replace rows $j_1, \ldots, j_r$ of the column by the number 1.
(3) If any entries of $K(k)$ remain empty, then replace them by the number 0. Return to Step 4. □

The following generalization of a notion introduced in Murphy *et al.* (2002, 2003) is central to our discussion.

Definition 11: The outcome $K_g(\Sigma)$ of Algorithm 3 is called the *generalized skeleton matrix* of the asynchronous machine Σ. □

Example 9: For the generalized stable state machine $\Sigma_{|s}$ of Example 6, we have $n = 3$ and $t = 1$, so that $n+t-1 = 3$. The matrix of one-step generalized stable

transitions is given by (13). The powers of this matrix can then be computed to be

$$R^2(\Sigma_{|s}) = \begin{pmatrix} \{aa, ac, ca, cc\} & N & N & N \\ ba & \{aa, ba, ca\} & \{aa, ba, ca\} & \{aa, ba, ca\} \\ bc & \{ac, bc, cc\} & \{ac, bc, cc\} & \{ac, bc, cc\} \\ \{ab, bb, cb\} & \{ab, bb, cb\} & \{ab, bb, cb\} & \{ab, bb, cb\} \end{pmatrix},$$

$$R^3(\Sigma_{|s}) = \begin{pmatrix} \left\{\begin{matrix} aaa, aac, aca, acc \\ caa, cac, cca, ccc \end{matrix}\right\} & N & N & N \\ \left\{\begin{matrix} aba, baa, bba, bca \\ cba \end{matrix}\right\} & \left\{\begin{matrix} aaa, aba, aca, baa \\ bba, bca, caa, cba \\ cca \end{matrix}\right\} & \left\{\begin{matrix} aaa, aba, aca, baa \\ bba, bca, caa, cba \\ cca \end{matrix}\right\} & \left\{\begin{matrix} aaa, aba, aca \\ baa, bba, bca, caa, cba \\ cca \end{matrix}\right\} \\ \left\{\begin{matrix} abc, bac, bbc, bcc \\ cbc \end{matrix}\right\} & \left\{\begin{matrix} aac, abc, acc, bac \\ bbc, bcc, cac, cbc \\ ccc \end{matrix}\right\} & \left\{\begin{matrix} aac, abc, acc, bac \\ bbc, bcc, cac, cbc \\ ccc \end{matrix}\right\} & \left\{\begin{matrix} aac, abc, acc, bac \\ bbc, bcc, cac, cbc \\ ccc \end{matrix}\right\} \\ \left\{\begin{matrix} aab, abb, acb \\ bab, bbb, bcb, cab \\ cbb, ccb \end{matrix}\right\} & \left\{\begin{matrix} aab, abb, acb, bab \\ bbb, bcb, cab, cbb \\ ccb \end{matrix}\right\} & \left\{\begin{matrix} aab, abb, acb, bab \\ bbb, bcb, cab, cbb \\ ccb \end{matrix}\right\} & \left\{\begin{matrix} aab, abb, acb, bab \\ bbb, bcb, cab, cbb \\ ccb \end{matrix}\right\} \end{pmatrix}.$$

Using these matrices, the matrix $\Gamma(\Sigma)$ is derived through (14). Finally, applying Algorithm 3 to $\Gamma(\Sigma)$, we obtain the generalized skeleton matrix

$$K_g(\Sigma) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{16}$$

□

We have reached a critical step stone on our way toward the solution of the model matching problem.

Proposition 7: *Let Σ be an asynchronous machine with the generalized skeleton matrix $K_g(\Sigma)$, and let x^i and x^j be two generalized states of Σ. Then, the following are equivalent.*

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

Proof: Let s be the generalized stable recursion function of the asynchronous machine Σ. Note that the matrix $K(1)$ obtained at the end of Step 2 of Algorithm 3 is characterized by the following properties: (α) it has 1 in position (i,j) if and only if $\Sigma_{|s}$ has a deterministic transition from the state x^j to the state x^i; and (β) an input character u appears in positions $i_1, \ldots, i_q$ of column j if and only if the pair (x^j, u) is a critical race pair with the outcomes x^{i_1}, $x^{i_2}, \ldots, x^{i_q}$.

Clearly, case (α) is valid if and only if there is an input value u such that $x^i = s(x^j, u)$. It is easy to see that the Proposition is valid for entries of $K_g(\Sigma)$ generated by case (α). The remaining part of the proof addresses case (β).

Assume then that there is a feedback trajectory $S_0', S_1', \ldots S_p'$ from x^j to x^i. Define a new feedback trajectory $S_0, \ldots, S_p$ from x^j to x^i as follows. Set $S_0 := S_0'$; obtain S_1 by removing from S_1' all pairs that include generalized states not in the set $s[S_0]$; obtain S_2 by removing from S_2' all pairs that include generalized states not in the set $s[S_1]$; and so on. In general, assuming that S_k has been derived in this way, obtain S_{k+1} by removing from S_{k+1}' all pairs that include generalized states not in the set $s[S_k]$, $k = 1, \ldots, p-1$. Then, $\{S_0, \ldots, S_p\}$ still forms a feedback trajectory from x^j to x^i.

By the definition of a feedback trajectory, $S_0 = \{(x^j, u)\}$ for some input character $u \in A$, and $s[S_p] = x^i$. Further, let $\Pi_x[S_k] = \{x^{i(1,k)}, \ldots, x^{i(q(k),k)}\}$ be the corresponding set of generalized states. Then, using again the definition of a feedback trajectory, there are input characters $u(1,k), \ldots, u(q(k),k) \in A$ such that $(x^{i(r,k)}, u(r,k)) \in S_k$ and $s(x^{i(r,k)}, u(r,k)) \in \Pi_x[S_{k+1}]$, $r = 1, \ldots, q(k)$, $k = 0, \ldots, p$. Setting $k = p$ and recalling that $s[S_p] = x^i$, it follows that, for each one of the generalized states $x^{i(1,p)}, \ldots, x^{i(q(p),p)}$, there is an input character that takes that state to the generalized

state x^i in a one step deterministic stable transition. By Step 5 of Algorithm 3, we conclude that the matrix $K_g(\Sigma)$ has an entry of 1 in positions $(i, i(1,p)), \ldots, (i, i(q(p),p)$.

To continue the proof inductively, assume that $K_g(\Sigma)$ has an entry of 1 in positions $(i, i(1,k)), \ldots, (i, i(q(k),k)$ for some $k \in \{1, \ldots, p\}$. In view of the fact that $\{x^{i(1,k)}, \ldots, x^{i(q(k),k)}\} = s[S_{k-1}]$ by construction, it follows by Step 5 of Algorithm 3 that the matrix $K_g(\Sigma)$ has an entry of 1 in positions $(i, i(1,k-1)), \ldots, (i, i(q(k-1),k-1))$. As this is true for every $k = 1, \ldots, p$, and since $S_0 = \{(x^j, u)\}$, we conclude that there is an entry of 1 in position (i,j) of the matrix $K_g(\Sigma)$. Thus, (i) implies (ii).

Conversely, assume that there is an entry of 1 in position (i,j) of the generalized skeleton matrix $K_g(\Sigma)$. We build a feedback trajectory from x^j to x^i as follows. Consider first the matrix $K(1)$ of Algorithm 3. By construction, an entry of 1 in a position (i,j) of $K(1)$ is equivalent to the existence of a deterministic transition from the generalized state x^j to the generalized state x^i. This implies the existence of a feedback trajectory from x^j to x^i.

To consider the other possibilities, refer to Step 5 of Algorithm 3. Let k_0 be the first value of k at which the condition of Step 5(a) is satisfied. Then, $K(k_0) = K(1)$, and, consequently, all existing entries of 1 in the matrix $K(k_0)$ designate deterministic transitions. By the construction of $J(u)$ in Step 5(d) of Algorithm 3, this implies that, if there is an entry of 1 in row j of $J(u)$, then there is a deterministic transition from every one of the states $x^{i_1}, \ldots, x^{i_q}$ to the state x^j. In other words, there are input strings $u(1, i_1), \ldots, u(1, i_q)$ such that

$$x^j = s\big(x^{i_1}, u(1, i_1)\big) = s\big(x^{i_2}, u(1, i_2)\big) = \cdots = s\big(x^{i_q}, u\big(1, i_q\big)\big).$$

In view of Step 5(a) of Algorithm 3, we have $s(x^{k_0}, u) = \{x^{i_1}, \ldots, x^{i_q}\}$. A brief contemplation indicates that the last two relations induce a feedback trajectory from x^{k_0} to x^j. Applying these arguments to each entry of column k_0 of the matrix $K(k_0)$, it follows that the following holds for the matrix $K(k_0+1)$: If an entry of 1 appears in position (i,j) of $K(k_0+1)$, then there is a feedback trajectory from x^{k_0} to x^i.

The reasoning of the previous paragraph can be applied recursively at every cycle of Algorithm 3. This leads to the conclusion that, if an entry of 1 appears in position (i,j) of the matrix $K_g(\Sigma)$, then there is a feedback trajectory from x^j to x^i. Finally, referring to Step 3(b), assume that $\alpha \geq 2$ and $K_\alpha(\Sigma) = K_{\alpha-1}(\Sigma)$. A slight reflection will show that any strings of A^+ present in $K_\alpha(\Sigma)$ indicate critical races whose outcomes cannot be guided toward a single state by a state feedback controller. Hence, these entries can be replaced by zeros, and we conclude that (ii) implies (i). □

The generalized skeleton matrix plays a critical role in our discussion, reminiscent of the role played by the skeleton matrix in Murphy *et al.* (2002, 2003) and in Geng and Hammer (2004, 2005). Combining Propositions 4, 6, and 7, we reach the following conclusion, which forms one of the main results of this paper.

Theorem 2: *Let Σ be an asynchronous machine with the generalized skeleton matrix $K_g(\Sigma)$, and let x^i and x^j be two generalized states of Σ. Then, the following are equivalent.*

(a) *There is a feedback controller that takes Σ from x^j to x^i in semi-fundamental mode operation.*
(b) *The (i,j) entry of $K_g(\Sigma)$ is* 1. □

Theorem 2 provides a complete characterization of the potential for controlling an asynchronous machine. One simple consequence of the Theorem relates to the problem of stopping infinite cycles. Consider an asynchronous machine Σ with n states and $t \geq 1$ infinite cycles, and let $K_g(\Sigma)$ be its generalized skeleton matrix. A brief examination of Theorem 2 and of the structure of the generalized skeleton matrix leads to the following conclusion. All infinite cycles of Σ can be stopped with a deterministic outcome if and only if each one of the columns $n+1, n+2, \ldots, n+t$ of $K_g(\Sigma)$ contains at least one entry of 1 in its first n rows. We conclude this section with the following.

Proposition 8: *Algorithm* 3 *has polynomial complexity.*

Proof: Recall that, for a machine with n states and t infinite cycles, the generalized stable reachability matrix $\Gamma(\Sigma)$ is an $(n+t) \times (n+t)$ matrix. Tracing Algorithm 3, we can see that Step 1 has at most $(n+t)^2$ operations; Step 2 has at most $2(n+t)^2$ operations; Step 3 has at most $2(n+t)^2$ operations; Step 4 has at most $(n+t)$ checks; and Step 5 has at most $2(n+t)^2$ operations. Thus, the total number of operations in each cycle of Algorithm 3 is not more than $7(n+t)^2 + (n+t)$. Applying Lemma 4 to the generalized stable state machine $\Sigma_{|s}$, it follows that the longest chain of transitions without repetitions consists of $(n+t-1)$ steps (see also the proof of Murphy *et al.* (2003, Lemma 3.9)). Thus, Algorithm 3 terminates after at most $(n+t)$ cycles. Combining the last two sentences, we conclude that Algorithm 3 has a maximum of $7(n+t)^3 + (n+t)^2$ operations, and, consequently, has polynomial complexity. □

5. Corrective controllers

In the present section, we consider the construction of controllers that solve the model matching problem.

Let $\Sigma = (A, X, f)$ be an input/state asynchronous machine controlled by a controller C according to figure 1. Recall that Σ_c denotes the machine induced by the closed loop system. The stable state machine induced by Σ_c is denoted by $\Sigma_{c|s}$. Let $\Sigma' = (A, X, s')$ be a stable-state input/state machine having the same input set and the same state set as the machine Σ, and serving as the model. To address the model matching problem, we seek a controller C for which $\Sigma_{c|s} = \Sigma'$ for all initial conditions.

Now, since the model Σ' has the input alphabet A, this must also be the input alphabet of the composite system Σ_c, i.e., the external input set of figure 1. Accordingly, the input set of the controller C is the cross product $A \times X$, encompassing the two inputs of C: the external input of figure 1 and the state x of the controlled machine Σ. Further, as depicted in figure 1, the output of C serves as the input of Σ; thus, it is natural to let A be the output alphabet of C. Denoting by Ξ be the state set of C, we can write $C = (A \times X, A, \Xi, \xi_0, \phi, \eta)$, where ϕ and η are, respectively, the recursion function and the output function of C, and ξ_0 is the initial state of C. The composite system Σ_c then has the input set A and the state set $X \times \Xi$. Let f_c and h_c be, respectively, the recursion function and the output function of Σ_c. Considering that the output of figure 1 is the output of Σ, it follows that the output set of Σ_c is X, so that $\Sigma_c = (A, X, X \times \Xi, (x_0, \xi_0), f_c, h_c)$ and the output function satisfies $h_c(x, \xi, v) = x$ for all $(x, \xi, v) \in X \times \Xi \times A$. Introducing the standard projection Π_x: $X \times \Xi \times A \to X$: $\Pi_x(x, \xi, v) := x$, we have $h_c = \Pi_x$.

Now, let γ be the stable recursion function of Σ_c, i.e., the recursion function of $\Sigma_{c|s}$. Since C solves the model matching problem, i.e., $\Sigma_{c|s} = \Sigma'$, it follows that, for every valid pair (x, v) of Σ', there is a state $\xi \in \Xi$ for which $h_c\gamma(x, \xi, v) = s'(x, v)$. Using the equality $h_c = \Pi_x$, this becomes

$$\Pi_x\gamma(x, \xi, v) = s'(x, v).$$

An alternative representation of $\Sigma_{c|s}$ is important to our discussion. Note that, since the controller C has a fixed initial state ξ_0, its next stable state is determined by the external input of figure 1 and by the stable state of Σ. Thus, $\Sigma_{c|s}$ can be represented by a stable recursion function $s_c : X \times A \to X : (x, u) \mapsto s_c(x, u)$. When $\Sigma_{c|s}$ is stably equivalent to Σ', then $s_c = s'$. We will use this alternative representation of $\Sigma_{c|s}$ most often.

Further, let $X = \{x^1, \ldots, x^n\}$ be the state set of the asynchronous machine Σ, and assume that Σ has $t > 0$ infinite cycles. The generalized state set of Σ is then $\tilde{X} = \{x^1, \ldots, x^{n+t}\}$. For a cycle state $z \in \tilde{X}$ let $\rho(z) = \{x^{k_1}, x^{k_2}, \ldots, x^{k_p}; a\}$ be the corresponding infinite cycle, and let $\Pi_x\rho(z) = \{x^{k_1}, x^{k_2}, \ldots, x^{k_p}\} \subset X$ be the states of the infinite cycle. Define

$$\varphi(z) = \begin{cases} z & \text{if } z \text{ is a regular state,} \\ \Pi_x\rho(z) & \text{if } z \text{ is a cycle state.} \end{cases}$$

Note that $\varphi(z)$ yields a (regular) state or a set of (regular) states of Σ; no cycle states appear in the result of φ.

The following statement characterizes the basic technical requirements for the existence of a solution of the model matching problem (compare to Murphy *et al.* (2003, Theorem 4.3). Its proof describes the construction of a model matching controller.

Theorem 3: *Let $\Sigma = (A, X, f)$ be an input/state machine with stable recursion function s, and let $z^1, \ldots, z^q$ be generalized states of Σ. Let $U_1, \ldots, U_q$ be sets of input characters for which the product sets $z^1 \times U_1$, $z^2 \times U_2, \ldots, z^q \times U_q$ are all disjoint. For each $i = 1, \ldots, q$, let z'^i be a state of Σ that can be reached from z^i via a feedback trajectory. Then, there is a controller C that makes $\Sigma_{c|s}$ stably equivalent to a stable-state machine $\Sigma' = (A, X, s')$, whose stable recursion function s' satisfies*

(i) *$s'[\varphi(z^i), U_i] = z'^i$ for all $i = 1, \ldots, q$, and*

(ii) *$s'(x, t) = s(x, t)$ for all pairs $(x, t) \in X \times A \setminus \bigcup_{i=1,\ldots,q} z^i \times U_i$ for which s is defined.*

Moreover, Σ' is deterministic and free of infinite cycles, and the closed loop system Σ_c is well posed and operates in semi-fundamental mode.

Proof: Under the given assumptions, we construct a controller that satisfies requirements (i) and (ii). First, note that the situation here is similar to that of Proposition 6, except that instead of the single transition $x^j \to x^i$ that appears there, the controller here must induce the set of transitions $z^i \to z'^i$, $i = 1, \ldots, q$. However, each one of the transitions $z^i \to z'^i$ satisfies the requirements of Proposition 6. Thus, our present controller can be assembled from q controllers of the kind derived in the proof of Proposition 6, as follows.

For each integer $i = 1, \ldots, q$, follow the construction described in the proof of Proposition 6 to obtain the controller $C(z^i, z'^i, U_i)$. Assemble the controller

$$C = C(z^1, z'^1, U^1)\mathcal{V}C(z^2, z'^2, U^2)\mathcal{V}\cdots\mathcal{V}C(z^q, z'^q, U^q),$$

which operates as follows.

(i) If the controllers $C(z^1, z'^1, U_1)$, $C(z^2, z'^2, U_2), \ldots, C(z^q, z'^q, U_q)$ are all in their initial states, then $C = C(z^1, z'^1, U_1)$.

(ii) If any of the controllers $C(z^1, z'^1, U_1)$, $C(z^2, z'^2, U_2), \ldots, C(z^q, z'^q, U_q)$ has reached its

transition state (see proof of Proposition 6), then $C = C(z^r, z'^r, U_r)$, where r is the smallest integer $k \in \{1, \ldots, q\}$ for which $C(z^k, z'^k, U_k)$ has reached its transition state (note that $z^1, \ldots, z^q$ are not necessarily distinct states of Σ).

(iii) If one of the controllers $C(z^1, z'^1, U_1)$, $C(z^2, z'^2, U_2), \ldots, C(z^q, z'^q, U_q)$, say the controller $C(z^i, z'^i, U_i)$, is in a state other than its initial state or its transition state, then $C = C(z^i, z'^i, U_i)$.

The fact that the sets $z^1 \times U_1$, $z^2 \times U_2, \ldots, z^q \times U_q$ are all disjoint implies that, at any given time, only one of the controllers $C(z^1, z'^1, U_1)$, $C(z^2, z'^2, U_2), \ldots, C(z^q, z'^q, U_q)$ can be in a state other than its initial state or its transition state.

In view of this construction, the stable recursion function s' of the closed loop system Σ_c satisfies $s'[z^i, U_i] = z'^i$ for all $i = 1, \ldots, q$ and $s'(x, t) = s(x, t)$ for all $(x, t) \in X \times A \setminus \bigcup_{i=1,\ldots,q} z^i \times U_i$. By Definition 4 of the generalized recursion function, the first set of equalities yields $s'[\varphi(z^i), U_i] = z'^i$ for all $i = 1, \ldots, q$. Thus, (i) and (ii) are valid. Finally, noting that the stable recursion function s is defined only on pairs having a next stable state, our proof concludes. □

Note that, when z^i is a cycle state, condition (i) of Theorem 3 implies that the stable recursion function s' of the closed loop system Σ_c must be constant over the states included in the infinite cycle represented by z^i. This requirement is, however, of little practical impact, since it refers only to infinite cycles that form a persistent status of the closed loop system. The requirement does not apply to infinite cycles of Σ that are activated only transiently, when Σ passes through an infinite cycle on its way from one stable combination to another. Under normal circumstances, infinite cycles do not appear as a persistent status of Σ_c; in fact, the elimination of persistent infinite cycles is one of the objectives of using a corrective controller.

6. The model matching problem

We turn now to the solution of the model matching problem for input/state asynchronous machines with infinite cycles. Consider an asynchronous machine Σ having the state set $X = \{x^1, \ldots, x^n\}$ and $t \geq 1$ infinite cycles. The generalized state set $\tilde{X}$ of Σ consists then of $n + t$ generalized states $x^1, \ldots, x^{n+t}$. The generalized skeleton matrix $K_g(\Sigma)$ is then an $(n + t) \times (n + t)$ matrix. In view of Proposition 7, a non-zero entry in row i of column j of $K_g(\Sigma)$ indicates the existence of a feedback trajectory from the generalized state x^j to the generalized state x^i.

One of the objectives of a model matching controller C is to eliminate persistent infinite cycles. Thus, once the controller C has been activated, there is no more interest in transitions of Σ that either start or end at generalized stable combinations with cycle states. Now, transitions that start at generalized stable combinations with cycle states are represented by the last t columns of $K_g(\Sigma)$, while transitions that terminate at generalized stable combinations with cycle states are represented by the last t rows of $K_g(\Sigma)$. Thus, for design purposes, the last t rows and the last t columns of $K_g(\Sigma)$ can be deleted. This leads to the following notion.

Definition 12: Let Σ be an asynchronous machine with n states and t infinite cycles, and let $K_g(\Sigma)$ be its generalized skeleton matrix. The skeleton matrix $K(\Sigma)$ is obtained by deleting the last t rows and the last t columns of $K_g(\Sigma)$. □

Note that, although all rows and columns corresponding to cycle states were removed, the entries of the skeleton matrix may still include transitions that pass transiently through infinite cycles. The only transitions that were eliminated are the ones that start or end at infinite cycles.

Example 10: For the generalized skeleton matrix $K_g(\Sigma)$ of (16), we have

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

We are ready now to state the main result of the present section. (Given two $n \times n$ numerical matrices A and B, the inequality $A \geq B$ is interpreted entrywise, i.e., $A_{ij} \geq B_{ij}$ for all $i, j = 1, \ldots, n$.)

Theorem 4: *Let $\Sigma = (A, X, f)$ be an input/state asynchronous machine with the skeleton matrix $K(\Sigma)$, and let $\Sigma' = (A, X, s')$ be a stable-state input/state asynchronous machine with the skeleton matrix $K(\Sigma')$. Then, the two following statements are equivalent.*

(a) *There is a state-feedback controller C for which the closed loop system $\Sigma_{c|s}$ is stably equivalent to Σ', where Σ_c is well posed and operates in semi-fundamental mode.*

(b) *The skeleton matrices satisfy $K(\Sigma) \geq K(\Sigma')$.*

Theorem 4 provides a simple necessary and sufficient condition for the existence of a solution to the model matching problem for systems with infinite cycles. The proof of the Theorem, which is provided below, describes the construction of an appropriate controller C. The controller C transforms into unstable combinations all generalized stable combinations of Σ that do not correspond to stable combinations of the model Σ'.

In this way, the stable-state machine $\Sigma_{c|s}$ induced by the closed loop system becomes stably equivalent to Σ'. Being a stable state machine, the model Σ' has no infinite cycles, and, as a result, neither will the closed loop system Σ_c have any persistent infinite cycles. Still, within the closed loop, the machine Σ may pass transiently through infinite cycles on its way from one stable combination to another.

Proof of Theorem 4: Let $X=\{x^1,\ldots,x^n\}$ be the state set of Σ and of Σ'. Assume first that condition (i) of Theorem 4 is valid, i.e., that there is a controller C such that $\Sigma_{c|s}=\Sigma'$. Let s_c be the stable recursion function of Σ_c, and recall that s' is the stable recursion function of the model Σ'. The last equality means that $s_c(x,u)=s'(x,u)$ for all valid pairs $(x,u)\in X\times A$. Now, consider a pair of integers $i,j\in\{1,\ldots,n\}$ for which entry (i,j) of the skeleton matrix $K(\Sigma')$ is 1. Then, by the definition of the skeleton matrix, there is an input string $v\in A^+$ such that $s'(x^j,v)=x^i$; by the equivalence $\Sigma_{c|s}=\Sigma'$, we obtain $s_c(x^j,u)=s'(x^j,u)=x^i$. In other words, the state feedback controller C drives Σ from x^j to x^i in semi-fundamental mode operation. Consequently, by Theorem 2, entry (i,j) of the generalized skeleton matrix $K_g(\Sigma)$ must be 1. Since x^i and x^j are not cycle states, the latter implies that entry (i,j) of the skeleton matrix $K(\Sigma)$ is also 1. Briefly, if the (i,j) entry of $K(\Sigma')$ is 1, then so must be the (i,j) entry of $K(\Sigma)$. As all entries are either zero or one, it follows that $K(\Sigma)\geq K(\Sigma')$, and (i) implies (ii).

Conversely, assume that (ii) is valid, i.e., that $K(\Sigma)\geq K(\Sigma')$. Recall that s' is the stable recursion function of the model Σ' and that $X=\{x^1,\ldots,x^n\}$ serves as the state set of Σ' and of Σ. Let $\{(x^j,u^1(x^j)),\ldots,(x^j,u^{k_j}(x^j))\}$ be the set of all valid pairs of s' that include the state x^j, where $j\in\{1,\ldots,n\}$, and denote

$$x^{m(j,r)}:=s'(x^j,u^r(x^j)),\ r=1,\ldots,k_j,\ j=1,\ldots,n. \quad (17)$$

Note that relations (17) completely characterize the stable recursion function s' of the model. By definition of the skeleton matrix, (17) implies that entry $(m(j,r),j)$ of $K(\Sigma')$ is equal to 1. Since $K(\Sigma)\geq K(\Sigma')$ by assumption, and since an entry of $K(\Sigma)$ is either 0 or 1, we conclude that entry $(m(j,r),j)$ of $K(\Sigma)$ must also be equal to 1. In view of Definition 12, entry $(m(j,r),j)$ of the generalized skeleton matrix $K_g(\Sigma)$ is 1 as well. Invoking Proposition 7, we conclude that, for the machine Σ, there is a feedback trajectory from the state x^j to the state $x^{m(j,r)}$, $r=1,\ldots,k_j$, $j=1,\ldots,n$. Consequently, conditions (i) and (ii) of Theorem 3 are satisfied for Σ under the following assignments: $z^1\mapsto x^1$, $z^2\mapsto x^1,\ldots,z^{k_1}\mapsto x^1$, $z^{k_1+1}\mapsto x^2,\ldots,z^{k_1+k_2}\mapsto x^2,\ldots U_1\mapsto u^1(x^1)$, $U_2\mapsto u^2(x^1),\ldots,U_{k_1}\mapsto u^{k_1}(x^1)$, $U_{k_1+1}\mapsto u^1(x^2),\ldots,U_{k_1+k_2}\mapsto u^{k_2}(x^2),\ldots,z'^1\mapsto x^{m(1,1)}$, $z'^2\mapsto x^{m(1,2)},\ldots,z'^{k_1}\mapsto x^{m(1,k_1)}$, $z'^{k_1+1}\mapsto x^{m(2,1)},\ldots,z'^{k_1+k_2}\mapsto x^{m(2,k_2)},\ldots$. (Note that, since none of the states $x^1,\ldots,x^n$ is a cycle state, condition (i) of Theorem 3 reduces to (17)). Thus, considering that equalities (17) completely determine s', it follows by Theorem 3 that there is a controller C for which $\Sigma_{c|s}$ is stably equivalent to Σ'. Whence, (ii) implies (i), and our proof concludes. □

The construction of a controller that satisfies the requirements of Theorem 4 is described in the proof of Theorem 3. After constructing the controller, its state set can often be reduced by using well established techniques for the reduction of asynchronous machines (e.g., Kohavi (1970)). Further state reductions can be achieved by careful choice of the strings generated by the controller. Next, a comprehensive example.

7. Example

Consider the machine Σ of Example 2. Assume that Σ must be controlled to match the model $\Sigma'=(A,X,s')$, whose stable recursion function s' is given by table 2 below.

Noting that Σ' has no infinite cycles; its skeleton matrix is calculated as

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

Now, the skeleton matrix $K(\Sigma)$ of the machine Σ was calculated in Example 10, and a brief examination shows that indeed $K(\Sigma)\geq K(\Sigma')$ (in fact, $K(\Sigma)=K(\Sigma')$ in this case). In view of Theorem 4, this implies that there is a controller C for which the closed loop system Σ_c is stably equivalent to the model Σ'. To construct an appropriate controller C, we use the procedure outlined in the proof of Theorem 3. A comparison of Example 6 to table 2 shows that a suitable controller can be obtained by inducing the following two stable state transitions of $\Sigma_{|s}$: from x^1 to x^2 upon receiving the external input

Table 2. The model Σ'.

	a	b	c
x^1	x^2	x^3	x^1
x^2	x^2	x^3	x^3
x^3	x^2	x^3	x^3

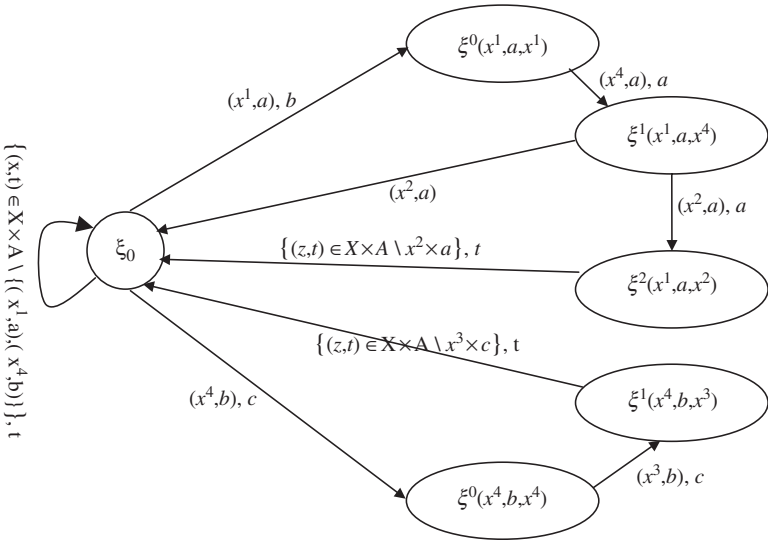


Figure 3. State flow diagram of the controller.

character a; and from the cycle state $x^4 = \{x^2, x^3; b\}$ to x^3 upon receiving the external input character b. A controller C that leads $\Sigma_{|s}$ through these two transitions, while leaving all other one-step transitions unchanged, solves the model matching problem in this case.

Adhering to the notation of the proof of Proposition 6, let $C(x^1, x^2, \{a\})$ be a controller that induces the first transition, and let $C(x^4, x^3, \{b\})$ be a controller that induces the second transition. As described in the proof, these controllers are constructed from corresponding feedback trajectories of Σ. In view of the proof of Theorem 3, our final controller is the combination $C(x^1, x^2, \{a\}) \vee C(x^4, x^3, \{b\})$.

We start with the feedback trajectory $\{S_0, S_1, S_2\}$ from x^1 to x^2 derived in Example 8. Recall that $S_0 = \{(x^1, b)\}$, $S_1 = \{(x^4, a)\}$, and $S_2 = \{(x^2, a)\}$. Let ϕ and η be the transition function and the output function, respectively, of the controller C. Using the notation of the proof of Proposition 6, we have $U(x^1) = \{a\}$. Letting ξ_0 be the initial state of the controller, set

$$\begin{aligned}
&\phi(\xi_0, (z,t)) := \xi_0 && \text{for all } (z,t) \in X \times A \setminus x^1 U(x^1),\\
&\phi(\xi_0, (x^1,u)) := \xi_1(x^1) && \text{for all } u \in U(x^1),\\
&\eta(\xi_0, (z,t)) := t && \text{for all } (z,t) \in X \times A,\\
&\eta(\xi_1(x^1), (x^1,t)) := a && \text{for all } t \in U(x^1).
\end{aligned}$$

The number of new controller states needed for this feedback trajectory is

$$P = \#\prod\nolimits_x S_0 + \#\prod\nolimits_x S_1 + \#\prod\nolimits_x S_2 = 1 + 1 + 1 = 3.$$

Implementing the construction outlined in the proof of Proposition 6, we obtain

$$\begin{aligned}
&\phi(\xi_1(x^1), x^1, a) := \xi^0(x^1, a, x^1),\\
&\eta(\xi^0(x^1,a,x^1), x, w) := b \quad \text{for all } (x,w) \in X \times A,\\
&\phi(\xi^0(x^1,a,x^1), x^4, a) = \xi^1(x^1, a, x^4),\\
&\eta(\xi^1(x^1,a,x^4), x, w) := a \quad \text{for all } (x,w) \in X \times A,\\
&\phi(\xi^1(x^1,a,x^4), x^2, a) = \xi^2(x^1, a, x^2),\\
&\eta(\xi^2(x^1,a,x^2), x, w) := a \quad \text{for all } (x,w) \in X \times A,\\
&\phi(\xi^2(x^1,a,x^2), (z,t)) := \xi^0 \quad \text{for all } (z,t) \in X \times A \setminus x^2 \times U(x^2),
\end{aligned}$$

where $U(x^2) = \{a\}$.

Next, we construct $C(x^4, x^3, \{b\})$ by a similar process. The feedback trajectory from x^4 to x^3 is the set $\{S_0, S_1\}$, where $S_0 = \{(x^4, c)\}$, $S_1 = \{(x^3, c)\}$. The number of new controller states for this feedback trajectory is

$$P = \#\prod\nolimits_x S_0 + \#\prod\nolimits_x S_1 = 1 + 1 = 2.$$

Further, we have $U(x^4) = \{b\}$. Again, using the construction outlined in the proof of Proposition 6, we obtain the following controller functions for this case:

$$\begin{aligned}
&\phi(\xi_0, (z,t)) := \xi_0 && \text{for all } (z,t) \in X \times A \setminus x^4 \times U(x^4),\\
&\phi(\xi_0, (x^4,u)) := \xi_1(x^4) && \text{for all } u \in U(x^4),\\
&\eta(\xi_0, (z,t)) := t && \text{for all } (z,t) \in X \times A,\\
&\eta(\xi_1(x^4), (x^4,t)) := b && \text{for all } t \in U(x^4),\\
&\phi(\xi_1(x^4), x^4, b) := \xi^0(x^4, b, x^4),\\
&\eta(\xi^0(x^4,b,x^4), x, w) := c && \text{for all } (x,w) \in X \times A,\\
&\phi(\xi^0(x^4,b,x^4), x^3, b) := \xi^1(x^4, a, x^3),\\
&\eta(\xi^1(x^4,b,x^3), x, w) := c && \text{for all } (x,w) \in X \times A,\\
&\phi(\xi^1(x^4,b,x^3), (z,t)) := \xi_0 && \text{for all } (z,t) \in X \times A \setminus x^3 \times U(x^3),
\end{aligned}$$

where $U(x^3) = \{c\}$. Applying standard reduction techniques to the controller $C(x^1, x^2, \{a\})$ $\mathcal{V}C(x^4, x^3, \{b\})$ yields the state flow diagram of a controller that solves the requisite model matching problem (see figure 3).

8. Conclusion

To summarize, the paper presents a control methodology that eliminates the effects of infinite cycles on the behaviour of asynchronous sequential machines. A critical aspect of the discussion is to guaranty that closure of the control loop introduces no hazards (critical races or infinite cycles) into the system. To use the methodology, one starts by selecting a desired behaviour to replace the one of the faulty machine, and by describing this desired behavior in the form of an asynchronous machine model. Then, the procedure outlined in §§ 5 and 6 leads (whenever possible) to the design of a state feedback controller that controls the given machine so as to match the desired model. As the desired model is faultless, the resulting controlled machine exhibits no ill effects from the infinite cycles of the original machine. The design procedure depends on several algorithms provided in the paper and can be carried out within any digital programming environment. Once a controller has been designed, it can be implemented as an asynchronous digital circuit and connected to the faulty machine.

References

G. Barrett and S. Lafortune, "Bisimulation, the supervisory control problem, and strong model matching for finite state machines", *Journal of Discrete Event Dynamic Systems*, 8, pp. 377–429, 1998.

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

M.D. Dibenedetto, A. Sangiovanni–Vincentelli and T. Villa, "Model matching for finite–state machines", *IEEE Transactions on Automatic Control*, 46, pp. 1726–1743, 2001.

S. Eilenberg, *Automata, Languages and Machines*, NY: Academic Press, 1974.

X.J. Geng and J. Hammer, "Asynchronous sequential machines: input/output control", *Proceedings of the 12th Mediterranean Conference on Control and Automation*, Kusadasi, Turkey, June 2004.

X.J. Geng and J. Hammer, "Input/output control of asynchronous sequential machines", *IEEE Trans. on Automatic Control*, 50, pp. 1956–1970, 2005.

J. Hammer, "On some control problems in molecular biology", *Proceedings of the IEEE Conference on Decision and Control*, pp. 4098–4103, December 1994.

J. Hammer, "On the modeling and control of biological signal chains", *Proceedings of the IEEE Conference on Decision and Control*, pp. 3742–3752, December 1995.

J. Hammer, "On the corrective control of sequential machines", *International Journal of Control*, 65, pp. 249–276, 1996a.

J. Hammer, "On the control of incompletely described sequential machines", *International Journal of Control*, 63, pp. 1005–1028, 1996b.

J. Hammer, "On the control of sequential machines with disturbances", *International Journal of Control*, 67, pp. 307–331, 1971.

D.A. Huffman, "The synthesis of sequential switching circuits", *J. Franklin Inst*., 257, pp. 161–190, 1954a.

D.A. Huffman, "The synthesis of sequential switching circuits", *J. Franklin Inst*., 257, pp. 275–303, 1954b.

D.A. Huffman, "The design and use of hazard-free switching networks", *Journal of the Association of Computing Machinery*, 4, pp. 47–62, 1957.

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

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

T.E. Murphy, "On the control of asynchronous machines with races", *IEEE Transactions on Automatic Control*, 48, pp. 1073–1081, 2003.

P.J.G. Ramadge and W.M. Wonham, "Supervisory control of a class of discrete event processes", *SIAM Journal of Control and Optimization*, 25, pp. 206–230, 1987.

J.G. Thistle and W.M. Wonham, "Control of infinite behavior of finite automata", *SIAM Journal on Control and Optimization*, 32, pp. 1075–1097, 1994.

S.H. Unger, "Hazards, critical races, and metastability", *IEEE Transactions on Computers*, 44, pp. 754–768, 1995.