International Journal of Control
Vol. 81, No. 12, December 2008, 1910–1929

Taylor & Francis
Taylor & Francis Group

State feedback control of asynchronous sequential machines with adversarial inputs

Jung-Min Yang[a] and Jacob Hammer[b]*

[a]*Department of Electrical Engineering, Catholic University of Daegu, 330 Kyongsan, Kyongbuk, Republic of Korea;*
[b]*Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA*

(*Received 5 October 2007; final version received 17 January 2008*)

The problem of controlling an asynchronous sequential machine in the presence of adversarial inputs is considered. Here, an adversarial input is an unknown and unauthorised input that attempts to interfere with the operation of the machine. The objective is to develop automatic state feedback controllers that counteract the effects of the adversarial input and restore desirable behaviour to the controlled machine. Necessary and sufficient conditions for the existence of such controllers are presented in terms of an inequality condition between two numerical matrices. Whenever a controller exists, an algorithm for its design is provided.

Keywords: asynchronous machines; control systems; feedback; disturbance rejection; computer security

1. Introduction

Unauthorised and adversarial input agents have, regretfully, become an unavoidable feature of modern day computing. These agents aim to interfere with the proper operation of computing systems, often attempting to subdue them to hostile objectives. The present paper addresses the question of how a computing system can be made immune to such attempts to interfere with its operation. Specifically, we propose to deploy automatic controllers that continually monitor a computing system and take corrective action whenever an adversarial input attempts to intercede. The paper presents necessary and sufficient conditions for the existence of such controllers; when a controller exists, an algorithm for its design is also provided.

In formal terms, the discussion revolves around asynchronous sequential machines that have two input signals: one input signal facilitates control of the machine, while the other is used by an adversarial agent (or by a disturbance) to interfere with the operation of the machine. The control diagram is shown in Figure 1, where Σ is the asynchronous sequential machine being controlled, and C is another asynchronous sequential machine that serves as a controller. The machine Σ has two inputs: u is the *control input* used to steer the machine to proper operation; and w is the *adversarial input* operated by an adversarial agent or by a disturbance. The purpose of the controller C is to counteract action at the input w and to endow the closed loop machine with desirable behaviour. Section 7 presents necessary and sufficient conditions for the existence of such a controller C, and Section 8 describes the structure of the controller. The closed loop machine described by the diagram is denoted by $\Sigma_c(v, w)$, where v is the external input of the closed loop machine. In our present discussion, C is a state feedback controller – it has access to the state of the machine Σ.

Our discussion is within the context of model matching. Let Σ' be an asynchronous sequential machine that describes the desired behaviour of the closed loop system. We refer to Σ' as the *model*. Of course, the model Σ' is not affected by the adversarial input w – it has no adversarial input; Σ' accepts only user commands as its input. The objective is to derive a controller C that drives the machine Σ so that, from a user's perspective, the closed loop system matches the behaviour of the model Σ', irrespective of actions taken at the adversarial input. In other words, we would like to have $\Sigma_c(\bullet, w) = \Sigma'(\bullet)$ for all adversarial input values w. This is the *perturbed model matching problem*. In this paper, we derive necessary and sufficient conditions for the existence of such a controller C; when C exists, we provide an algorithm for its construction.

Recall that an asynchronous sequential machine has two kinds of states: *stable states*, namely, states at which the machine dwells indefinitely with its present input value, and *transient states*, or states the machine passes transiently along its way to the next stable state. Only stable states are perceivable by the machine's

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

ISSN 0020–7179 print/ISSN 1366–5820 online

DOI: 10.1080/00207170801930225
http://www.informaworld.com

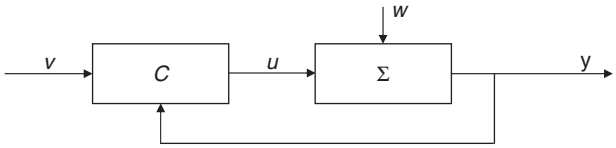

Figure 1. A closed loop configuration.

user; transient states are traversed by the machine very quickly (ideally, in zero time), and are imperceptible to the user. Thus, when eliminating the effects of an adversarial input, it is only necessary to eliminate the effects on stable states; effects on transient states are unnoticeable and, therefore, inconsequential.

The ability to control an asynchronous machine Σ so as to eliminate the effects of adversarial inputs and match a specified model depends on certain features of reachability and detectability introduced in §4. The essence of these features is condensed in §6 and 7 into a numerical matrix of zeros and ones, called a *skeleton matrix*. The skeleton matrix characterises the possibilities of controlling the machine Σ in the presence of an adversarial input. More specifically, the skeleton matrix characterises those aspects of the performance of Σ that can be preserved by an automatic controller despite activity at the adversarial input. In §7, we show that the perturbed model matching problem is solvable if and only if the skeleton matrix satisfies a certain numerical inequality; compare to Murphy, Geng and Hammer (2002, 2003).

The present paper deals with the control of asynchronous sequential machines utilising the formalism of Murphy et al. (2002, 2003), Geng and Hammer (2004, 2005), and Venkatraman and Hammer (2006a,b,c). Of course, asynchronous sequential machines are a topic within the general area of discrete mathematics. To a large extent, our terminology and notation follow Eilenberg (1974). Studies dealing with other aspects of the control of discrete event systems can be found in Ramadge and Wonham (1987), Hammer (1994, 1995, 1996a,b, 1997), Dibenedetto, Saaldanha and Sangiovanni–Vincentelli (1994), Thistle and Wonham (1994), Barrett and Lafortune (1998), the references cited in these papers, and others. It seems, however, that these publications do not address issues that are peculiar to the function of asynchronous machines, such as the avoidance of critical races and the distinction between stable states and transient states.

An important aspect of the operation of asynchronous sequential machines is fundamental mode operation; e.g., Kohavi (1970). Under fundamental mode operation, the input variables of an asynchronous machine are kept constant while the machine undergoes state transitions. Fundamental mode operation comes to guarantee deterministic behaviour. Indeed, if an input value is changed while a machine undergoes state transitions, then the state at which the input change occurs becomes unpredictable; this may result in an unpredictable outcome.

In the case of Figure 1, fundamental mode operation means that (i) the controller C must be in a stable state while Σ undergoes transitions, and (ii) the machine Σ must be in a stable state while C undergoes transitions. All systems considered in this paper are designed to operate in fundamental mode.

The paper is organised as follows. Section 2 reviews and expands the basic notation and framework of our discussion, and Section 3 introduces adversarial inputs and examines some of their potential effects. Two notions that are critical to the solution of the adversarial model matching problem – the notions of reachability and detectability – are discussed in Section 4. Section 5 introduces a test that determines whether or not an adversarial action can be counteracted. Necessary and sufficient conditions for the existence of a controller that solves the perturbed model matching problem are derived in Sections 6 and 7, while the structure of the controller is described in Section 8. The paper concludes in Section 9 with a comprehensive example.

2. Notation and basics

Let A be a finite non-empty alphabet, let A^* be the set of all finite strings of characters of A, and let A^+ be the set of all non-empty strings in A^*. To simplify our notation later, 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 $w := w_1 w_2$ obtained by appending w_2 to the end of w_1. A partial function $f: S_1 \to S_2$ is a function whose domain is a subset of S_1.

Consider an asynchronous sequential machine with two inputs: a control input through which the machine is operated and an adversarial input that attempts to interfere with the operation of the machine. We represent such a machine by a sextuple $\Sigma = (A \times B, Y, X, x_0, f, h)$, where A is the control input alphabet, B is the adversarial input alphabet, Y is the output alphabet, X is a set of n states, and x_0 is the initial state of the machine; $f: X \times A \times B \to X$ (the *recursion function*) and $h: X \to Y$ (the *output function*) are partial functions. The machine Σ operates recursively according to

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

Here, $u_0, u_1, u_2, \ldots$ is the *control input sequence*, while $w_0, w_1, w_2, \ldots$ is the *adversarial input sequence*; $x_0, x_1, x_2, \ldots$ is the sequence of the states through which the machine passes, and $y_0, y_1, y_2, \ldots$ is the sequence of output values. The integer k represents the *step counter*; it advances by one upon any change of the machine's inputs or state. Having selected the Moore representation for Σ, the output function h does not depend on the input variables u and w; see Kohavi (1970).

Example 1: The following represents an asynchronous machine Σ with adversarial input. Here, the control input alphabet is $A=\{a,b\}$; the adversarial input alphabet is $B=\{\alpha,\beta\}$; the state set is $X=\{x^1,x^2,x^3\}$. We assume that the state of the machine is also its output, so that the output function h is the identity function. The recursion function f of Σ is described by Figure 2. Alternatively, the transition function f can be characterised by Table 1.

A triplet (x,u,w) is a *valid combination* of Σ if it is included in the domain of the function f. Occasionally, we use a single character for the two inputs, as in $\alpha_k=(u_k,w_k)$, $k=0,1,2,\ldots$ The input sequence $\alpha_0,\alpha_1,\alpha_2,\ldots$ is permissible if all pairs $(x_0,\alpha_0),(x_1,\alpha_1),(x_2,\alpha_2),\ldots$ are valid.

Consider an input string with repeated characters, say *aaabbcccc*. In compressed notation, the repetitions are omitted, so our string becomes simply *abc*. This notation conforms to the way the string is applied in practice – the first input character is kept constant for the first three steps, then the second character is kept constant for two steps, and, finally, the last input character is kept constant for four steps.

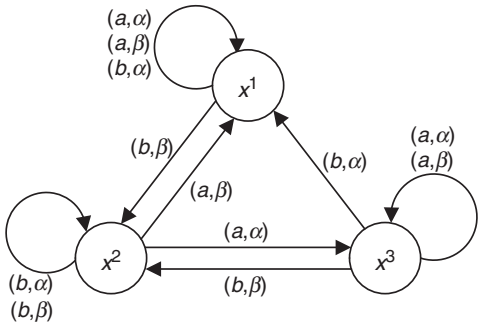

Figure 2. A transition diagram.

Table 1. A transition table.

	(a,α)	(a,β)	(b,α)	(b,β)
x^1	x^1	x^1	x^1	x^2
x^2	x^3	x^1	x^2	x^2
x^3	x^3	x^3	x^1	x^2

The machine Σ is an *input/state machine* when its output is the state, namely, when $Y=X$ and $h(x)=x$ for all $x\in X$. Then, $y_k=x_k$ for all $k=0,1,2,\ldots,$ and the machine is described by the recursion

$$\Sigma: x_{k+1}=f(x_k,u_k,w_k). \tag{2}$$

An input/state machine is represented by a quadruple $\Sigma=(A\times B,X,x_0,f)$. The present paper deals with the control of asynchronous input/state machines.

A valid triplet (x,u,w) is a *stable combination* if $x=f(x,u,w)$, namely, if the state x is a fixed point of the recursion function f. An asynchronous machine lingers at a stable combination until a change occurs at one of its inputs. A triplet (x,u,w) that is not a stable combination is a *transient combination*.

A transient triplet (x,u,w) initiates a chain of transitions $x_1=f(x,u,w), x_2=f(x_1,u,w),\ldots,$ where the input characters u and w are kept fixed while the states change. This chain of transitions may or may not terminate. If it terminates, then there is an integer $q\geq 1$ for which $x_q=f(x_q,u,w)$, i.e., (x_q,u,w) is a stable combination. Then, x_q is the next stable state of x with the input pair (u,w). If the chain of transitions does not terminate, then the triplet (x,u,w) is part of an infinite cycle. In this paper, we restrict our attention to machines that have no infinite cycles. Thus, in our case, every valid triple (x,u,w) has a next stable state. For future reference, it is convenient to record this fact.

Lemma 1: *In an asynchronous machine without infinite cycles, every valid combination has a next stable state.*

When operating an asynchronous machine, one has to be careful to prevent situations where two or more variables change value at the same time. A simultaneous change of two or more variables may cause an asynchronous machine to become unpredictable, e.g., Kohavi (1970). Thus, it is common to enforce a policy where only one variable is allowed to change value at any instant of time. When this policy is enforced, the machine Σ operates in *fundamental mode*. In practice, almost all asynchronous machines are operated in fundamental mode.

The asynchronous machine Σ of (2) involves three variables: the state x and the input variables u and w. In fundamental mode operation, not more than one of these variables can change value at any instant of time. This leads to the following.

Definition 1: Let $\Sigma=(A\times B,X,f)$ be an asynchronous input/state machine with the two input variables u and w. The machine Σ operates in *fundamental mode* if u and w change values only when Σ is in a stable combination, and then at most one at a time.

All the machines discussed in this paper operate in fundamental mode. Note that fundamental mode operation is not an overly restrictive requirement. An asynchronous machine reaches its next stable combination very quickly – ideally, in zero time; thus, it is rather unlikely that changes in u or in w will occur while Σ is in transition. Similarly, the probability that the two independent variables u and w will change values simultaneously is zero, since the two are not synchronised. Fundamental mode operation is most common in practice, and we adopt it as our mode of operation.

Transitions from one stable combination of Σ to another are governed by its *stable recursion function* s, which is defined as follows. For a valid triplet (x, u, w) of the machine Σ, let x' be the next stable state; the stable recursion function $s: X \times A \times B \to X$ is defined by the assignment $s(x, u, w) := x'$. As Σ has no infinite cycles, every valid triplet has a next stable state, and hence s is defined on all valid triplets. The stable recursion function s induces the *stable state machine* $\Sigma_{|s}$ of Σ, which is an asynchronous input/state machine given by $(A \times B, X, s)$. Here, the stable recursion function s replaces the recursion function f of Σ.

Consider an input string $\alpha := \alpha_0\alpha_1 \ldots \alpha_{m-1}$, where $\alpha_i \in A \times B$. Writing in terms of components, we have that $\alpha_i = (u_i, w_i)$, with $u_i \in A$ and $w_i \in B, i = 0, \ldots, m-1$. Assume that Σ is in a stable combination at the initial state x_0, when the input string α is applied. In fundamental mode operation, the first input value α_0 must remain fixed until Σ reaches its next stable state $x_1 := s(x_0, \alpha_0)$ (ideally, this transition is completed in zero time). Anytime thereafter, one of the input variables u or w can change, providing the next input pair α_1. These input values remain fixed until the next stable state $x_2 := s(s(x_0, \alpha_0), \alpha_1)$ is reached. This process continues until the last stable state $x_m := s(\ldots s(s(s(x_0, \alpha_0), \alpha_1), \alpha_2) \ldots, \alpha_{m-1})$ is reached. It is convenient to use the shorthand notation

$$s(x_0, \alpha) = s(\ldots s(s(s(x_0, \alpha_0), \alpha_1), \alpha_2) \ldots, \alpha_{m-1}), \quad \alpha \in (A \times B)^+. \tag{3}$$

When the machine $\Sigma = (A \times B, X, f)$ is connected in the closed loop of Figure 1, the output of the controller C serves as the control input of Σ. We therefore use A as the output alphabet of C. Also, the input variable v of the controller C in the figure is really intended to control the operation of Σ, so we use A as the alphabet of v as well. In addition to v, the controller C accepts the state X of Σ as another input, so the input set of C is $A \times X$. Letting Ξ be the state set, ϕ the recursion function, and η the output function of the controller, we can write $C = (A \times X, A, \Xi, \xi_0, \phi, \eta)$, where ξ_0 is the initial state of C.

In fundamental mode operation of Figure 1, only one of the asynchronous machines Σ and C can undergo transitions at any instant of time. Adapting Geng and Hammer (2005, Proposition 1.4) to our present case, we obtain the following.

Proposition 1: Let $\Sigma = (A \times B, X, f)$ and $C = (A \times X, A, \Xi, \xi_0, \phi, \eta)$ be asynchronous machines interconnected in the configuration of Figure 1. The configuration operates in fundamental mode if and only if all the following hold.

(i) C is in a stable combination while Σ undergoes transitions, and Σ is in a stable combination while C undergoes transitions.
(ii) The inputs u, w and v change only while Σ and C are in a stable combination, and then only one at a time.

Thus, the controller C must be designed so that it commences transitions only after verifying that Σ has reached a stable combination. Similarly, C must adopt a stable combination immediately prior to inducing a change at the input of Σ. The controller C designed in §8 below satisfies these requirements. Fundamental mode operation assures that all transitions of the composite system are unambiguous and deterministic.

3. Adversarial inputs and model matching

Our next objective is to discuss the implications of the adversarial input on the machine $\Sigma = (A \times B, X, f)$ of (2). The critical issue is, of course, the fact that the adversarial input character w_k is, in general, not specified. The only *a priori* information available about w_k is that it belongs to a specified subset $\nu \subset B$ called the *adversarial uncertainty*. Adding the adversarial uncertainty ν to the description of Σ, we obtain the quintuple $\Sigma = (A \times B, X, x_0, f, \nu)$.

When the adversarial uncertainty ν includes more than one character, the precise value of the adversarial input is not specified; this entails that the state of the machine Σ may not be known. Indeed, starting from the initial state x_0 and applying the initial control input value u_0, the state of Σ after the first step can be any member of the set

$$f[x_0 \times u_0 \times \nu] := \cup_{w \in \nu} f(x_0, u_0, w) \subset X.$$

When this set contains more than one state, one cannot predict the precise state of Σ after the first step. In such case, we are faced with the control of an asynchronous machine in the presence of uncertainty.

Recall that two asynchronous machines Σ_1 and Σ_2 are stably equivalent if the stable state machines $\Sigma_{1|s}$ and $\Sigma_{2|s}$ are equivalent; see Geng and Hammer (2004, 2005) for more details. Stably equivalent machines have equivalent functionality and are identical from a user's point of view. We are now ready to formally formulate the main topic of our discussion.

Problem 1: The perturbed model matching problem. Let $\Sigma = (A \times B, X, f, \nu)$ be an input/state asynchronous machine with an adversarial input, and let $\Sigma' = (A, X, s')$ be a stable-state input/state machine with no adversarial input. Find necessary and sufficient conditions for the existence of a controller C for which the closed loop machine $\Sigma_c(v, w)$ is stably equivalent to $\Sigma'(v)$ for all $w \in \nu$ and all $v \in A$. If such a controller exists, derive an algorithm for its design.

Example 2: A model machine Σ'. This input/state machine has the control input alphabet $A = \{a, b\}$ and the state set $X = \{x^1, x^2, x^3\}$. It is a stable state machine with the stable transition function s' given by the transition diagram in Figure 3; the model has no adversarial input. Alternatively, the stable transition function s' of Σ' can be described by Table 2.

Now, let s be the stable recursion function of the asynchronous machine $\Sigma = (A \times B, X, f, \nu)$. Assume that Σ is in a stable combination with the state x when the control input takes the value u. As Σ has no infinite cycles, it follows by Lemma 1 that there is a next stable state x' of Σ. The exact value of x' is usually impossible to predict, since the value w of the adversarial input is not known. All possible values of x' are given by the set

$$s^{\nu}(x, u) := s[x, u, \nu] = \{s(x, u, w) : w \in \nu\} \subset X. \quad (4)$$

Letting $P(X)$ be the family of all subsets of the state set X of Σ, we thus obtain the function $s^{\nu}: X \times A \to P(X)$ called the *perturbed stable recursion function* of Σ.

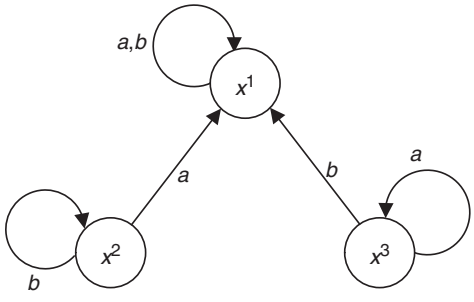

Figure 3. The model's transition diagram.

Table 2. The model's table of transitions.

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

4. Detectability and reachability

4.1 *Detectability*

As we have seen in Proposition 1, fundamental mode operation of the closed loop of Figure 1 requires the controller C to remain in a stable combination until the machine Σ has reached its next stable state. Consequently, it must be possible for C to determine whether or not Σ has reached its next stable state. We examine now the conditions under which the latter is possible.

To this end, let ν be the adversarial uncertainty of the machine Σ, and let $w \in \nu$ be the adversarial input character. Assume that Σ is in a stable combination with the state x, when the control input variable takes the value u. The machine Σ embarks then on a chain of state transitions given by

$$\theta(x, u, w) := \{x_1 := f(x, u, w), \\ x_2 := f(x_1, u, w), \ldots, x_{i(u,w)} := f(x_{i(u,w)-1}, u, w)\}, \quad (5)$$

where $x_{i(u,w)}$ is the stable state reached at the end. Clearly, the number of steps $i(u, w)$ and the state $x_{i(u,w)}$ depend on the adversarial input character $w \in \nu$. According to (4), we must have $x_{i(u,w)} \in s^{\nu}(x, u)$. The set

$$\theta[x, u, \nu] := \{\theta(x, u, w) : w \in \nu\} \quad (6)$$

includes all state strings that form chains of transitions consistent with the adversarial uncertainty ν, given that Σ starts from the state x with the control input character u. The following statement shows that a string in $\theta[x, u, \nu]$ includes no repeating states.

Lemma 2: *Let $\Sigma = (A \times B, X, f, \nu)$ be an asynchronous machine with no infinite cycles. Let x be a state of Σ and let u be a control input value. Then, a string $\theta \in \theta[x, u, \nu]$ includes no repeating states.*

Proof: Consider a string $\theta = \{x_1, x_2, \ldots, x_{i(u,w)}\} \in \theta[x, u, \nu]$. First, if $i(u, w) = 1$, then θ includes only one state, and clearly has no repeating states. Consider then the case where $i(u, w) > 1$. Assume, by contradiction, that $x_i = x_j$ for some integers $1 \le i < j \le i(u, w)$. Now, if $j = i(u, w)$, then the triple $(x_i, u, w) = (x_j, u, w) = (x_{i(u,w)}, u, w)$ is a stable combination. If view of (5), this

implies that θ terminates at step i, so that $i=j=i(u,w)$, a contradiction. Next, consider the case where $j<i(u,w)$. Using the recursion function of Σ, we get that $x_{i+1}=f(x_i,u,w)=f(x_j,u,w)=x_{j+1}, x_{i+2}=f(x_{i+1},u,w)=f(x_{j+1},u,w)=x_{j+2},\ldots,$ which implies that the machine Σ has an infinite cycle $x_i, x_{i+1}, x_{i+2},\ldots,x_j,x_{i+1},x_{i+2},\ldots$, contradicting our assumption. This concludes our proof. ■

The following notion underlies the controller's ability to determine whether or not the controlled machine Σ has reached its next stable state.

Definition 2: Let $\Sigma=(A\times B,X,f,\nu)$ be an input/state asynchronous machine, let $x\in X$ be a state of Σ, and let $u\in A$ be a control input character. Assume that Σ is in a stable combination with the state x, when the control input character changes to u. Then, the pair (x,u) is detectable if it is possible to determine from the state of Σ whether or not Σ has reached its next stable combination.

The following is a test that allows us to ascertain whether a given pair (x,u) is detectable.

Theorem 1: *Let $\Sigma=(A\times B,X,f)$ be an input/state asynchronous machine with an adversarial input, and let ε be the adversarial uncertainty. Let $x\in X$ be a state of Σ, let $u\in A$ be a control input character, and assume that Σ is in a stable combination with the state x when the control input character changes to u. Then, in the notation of* (4) *and* (6), *the following two statements are equivalent*:

(i) *the pair (x,u) is detectable*;
(ii) *states of the set $s^{\varepsilon}(x,u)$ appear only at the end of strings belonging to $\theta[x,u,\varepsilon]$*;

Proof: We use the notation of (5). Let $w\in\varepsilon$ be the active adversarial input character, and let $\theta(x,u,w)=\{x_0,x_1,x_2,\ldots,x_{i(u,w)}\}$ be the string of transitions initiated when the control input character switches to u. Assume that there is an integer $0\le j<i(u,w)$ for which $x_j\in s^{\varepsilon}(x,u)$. Since $j<i(u,w)$, it follows that (x_j,u,w) is a transient combination, as the chain of transitions continues after j. By the definition of the set $s^{\varepsilon}(x,u)$, the inclusion $x_j\in s^{\varepsilon}(x,u)$ implies that there is an adversarial input character $w'\in\varepsilon$ such that the triple (x_j,u,w') is a stable combination. Summarising, the state x_j forms a transient combination in the triple (x_j,u,w) while forming a stable combination in the triple (x_j,u,w'). Consequently, when Σ is in the state x_j, one cannot tell whether Σ is in a stable combination or not. In other words, when (ii) is invalid, so is (i).

Conversely, assume that a state $x'\in s^{\varepsilon}(x,u)$ can appear only at the end of a string belonging to $\theta(x,u,\varepsilon)$. Then, by the definition of $\theta(x,u,\varepsilon)$, the machine Σ is in a stable combination if and only if it is in a state $x'\in s^{\varepsilon}(x,u)$. In other words, one can determine from the state of Σ whether or not it has reached its next stable combination, and (x,u) is detectable. Thus, (ii) implies (i), and our proof concludes. □

Thus, for a detectable pair (x,u), a state feedback controller can determine whether or not Σ has reached its next stable state simply by checking whether the current state of Σ is a member of $s^{\varepsilon}(x,u)$. If it is, then Σ has reached its next stable state; if it is not, then Σ has not yet reached its next stable state. For pairs that are not detectable, it is not possible to determine from the current state whether Σ has reached its next stable combination.

Remark 1: Here, we assume that the state feedback controller provides information only about the current state of the machine Σ, and does not keep track of the entire trajectory of states traversed by Σ on its way to the current state. Controllers that keep track of the entire trajectory of Σ (i.e., controllers that record the 'burst' of Σ) have a more complex structure and will be discussed in a separate report.

4.2 *Reachability*

We turn now to an examination of the reachability features of asynchronous machines with adversarial input. First, we adapt to our present setting the following notion from Murphy et al. (2002, 2003).

Definition 3: Let $\Sigma=(A\times B,X,f,\nu)$ be an input/state asynchronous machine with adversarial input, having the state set $X=\{x^1,x^2,\ldots,x^n\}$. Let s be the stable transition function of Σ, let $u\in A$ be a control input character, and let $w\in B$ be an adversarial input character.

(i) The *one-step sample control transition matrix* $\gamma(\Sigma,w)$ of Σ is an $n\times n$ matrix whose (i,j) entry $\gamma_{ij}(\Sigma,w)$ consists of all control input characters $u\in A$ for which $x^j=s(x^i,u,w)$; if there are no such control input characters, then $\gamma_{ij}(\Sigma,w):=N$, where N is a character not in A or in B.
(ii) The *one-step sample adversarial transition matrix* $\lambda(\Sigma,u)$ of Σ is an $n\times n$ matrix whose (i,j) entry $\lambda_{ij}(\Sigma,u)$ consists of all adversarial input characters $w\in B$ for which $x^j=s(x^i,u,w)$; if there are no such adversarial input characters, then $\lambda_{ij}(\Sigma,u):=N$, where N is a character not in A or in B.

In order to keep an explicit record of the adversarial input character, we introduce the one-step matrix of stable transitions $\rho(\Sigma, w)$, where we add the prefix $w|$ to all entries of $\gamma(\Sigma, w)$, namely,

$$\rho_{ij}(\Sigma, w) := \{w|v : v \in \gamma_{ij}(\Sigma, w)\}, \quad i, j = 1, 2, \ldots, n.$$

Example 3: Consider the machine Σ of Example 1. The one-step matrices of stable transitions of Σ are

$$\rho(\Sigma, \alpha) = \begin{pmatrix} \{\alpha|a,\ \alpha|b\} & \{\alpha|N\} & \{\alpha|N\} \\ \{\alpha|N\} & \{\alpha|b\} & \{\alpha|a\}\ ) \\ \{\alpha|b\} & \{\alpha|N\} & \{\alpha|a\} \end{pmatrix},$$

$$\rho(\Sigma, \beta) = \begin{pmatrix} \{\beta|a\} & \{\beta|b\} & \{\beta|N\} \\ \{\beta|a\} & \{\beta|b\} & \{\beta|N\} \\ \{\beta|N\} & \{\beta|b\} & \{\beta|a\} \end{pmatrix}.$$

To work with the entries of $\rho(\Sigma, w)$, we define the projections Π_a: $B|(A^+ \cup N) \to B$ (projection onto the adversarial input value) and Π_c: $\mathrm{B}|(A^+ \cup N) \to (A^+ \cup N)$ (projection onto the control input value) by setting

$$\Pi_a w|u := \begin{cases} w & \text{if } u \neq N, \\ \varnothing & \text{else,} \end{cases}$$

and

$$\Pi_c w|u := u \quad \text{for all} \quad w|u \in B|(A^+ \cup N).$$

Next, given two sets of strings $s_1, s_2 \subset B|(A^+ \cup N)$, we define an operation $s_1 \vee s_2$ that is akin to the union of the two sets, with N being handled like the empty set it represents. Specifically, we delete from the union all elements $w|N$ for which w also appears with a nonempty control input string. Using $\setminus$ to denote the difference set, we have

$$s_1 \vee s_2 := [s_1 \cup s_2] \setminus s_N,$$

where s_N consists of all elements $w|N \in s_1 \cup s_2$ for which $[w|A^+] \cap [s_1 \cup s_2] \neq \varnothing$.

Recall that concatenation is an operation that combines two strings into one longer string. When dealing with pairs of strings of the form $w|u \in B|(A^+ \cup N)$, concatenation operates only on strings with the same adversarial input character, concatenating the control inputs and leaving the adversarial input character unchanged. Explicitly, given two strings $w_1|u_1, w_2|u_2 \in B|(A^+ \cup N)$, set

$$\mathrm{conc}(w_1|u_1, w_2|u_2) := \begin{cases} w_1|u_1u_2 & \text{if } w_1 = w_2 \text{ and both } u_1, u_2 \neq N, \\ w_1|N, w_2|\,N & \text{otherwise.} \end{cases}$$

For two subsets of strings $\sigma_1, \sigma_2 \subset B|(A^+ \cup N)$, the concatenation is defined by

$$\mathrm{conc}(\sigma_1, \sigma_2) := \vee_{s_1 \in \sigma_1,\, s_2 \in \sigma_2} \mathrm{conc}(s_1, s_2).$$

Further, we define an operation that resembles matrix multiplication. Let P and Q be two $n \times n$ matrices with entries in the set $B|(A^+ \cup N)$. The combination PQ is an $n \times n$ matrix with the entries

$$(PQ)_{ij} := \vee_{k=1,\ldots,n} \mathrm{conc}(P_{ik}, Q_{kj}), \quad i, j = 1, 2, \ldots, n.$$

Using this operation, we raise the one step matrix of stable transitions to the power of k, to get the matrix

$$\rho^k(\Sigma, w) = \rho^{k-1}(\Sigma, w)\rho(\Sigma, w), \quad k = 1, 2, \ldots.$$

By construction, the i,j entry of $\rho^k(\Sigma, w)$ consists of all strings of the form $w|u$ that take the machine Σ from a stable combination with the state x^i to a stable combination with the state x^j in exactly k steps; if no such transition is possible, then $u = N$.

Example 4: Continuing from Example 3, we obtain (omitting repeated characters)

$$\rho^4(\Sigma, \alpha) = \begin{pmatrix} \{\alpha|a, \alpha|aba, \alpha|baba, \alpha|ba, \alpha|abab, \alpha|ab, \alpha|bab, \alpha|b\} & \{\alpha|N\} & \{\alpha|N\} \\ \{\alpha|aba, \alpha|baba, \alpha|abab, \alpha|bab, \alpha|ab\} & \{\alpha|b\} & \{\alpha|ba, \alpha|a\} \\ \{\alpha|baba, \alpha|ba, \alpha|aba, \alpha|abab, \alpha|bab, \alpha|b, \alpha|ab\} & \{\alpha|N\} & \{\alpha|a\} \end{pmatrix}$$

$$\rho^4(\Sigma, \beta) = \begin{pmatrix} \{\beta|a, \beta|baba, \beta|aba, \beta|ba\} & \{\beta|abab, \beta|bab, \beta|ab, \beta|b\} & \{\beta|N\} \\ \{\beta|a, \beta|baba, \beta|aba, \beta|ba\} & \{\beta|abab, \beta|bab, \beta|ab, \beta|b\} & \{\beta|N\} \\ \{\beta|baba, \beta|ba, \beta|aba\} & \{\beta|abab, \beta|bab, \beta|b, \beta|ab, \beta|a\} & \{\beta|a\} \end{pmatrix}$$

The matrix

$$R(m, \Sigma, w) := \vee_{i=1,\ldots,m}\ \rho^i(\Sigma, w) \tag{7}$$

is the matrix of m stable transitions of Σ. It characterises all the transitions of Σ that can be accomplished in m or fewer stable steps. When we allow m to grow indefinitely, we obtain the extended matrix of stable transitions

$$R^*(\Sigma, w) := \vee_{i \geq 1}\ \rho^i(\Sigma, w),$$

which characterises all stable transitions of the machine Σ.

Using the fact that Σ has only n states, an argument similar to the one employed in Murphy et al. (2003, Proposition 3.9), yields the following.

Lemma 3: *Let $\Sigma=(A\times B, X, f)$ be an asynchronous machine with n states and an adversarial input. Let w be an adversarial input character of Σ, and let $R(m, \Sigma, w)$ be the matrix of m stable transitions of Σ. Then, the following two statements are equivalent for all integers $m\geq n-1$ and all $i,j=1,2,\ldots,n$.*

(i) *The entry $R_{ij}(m, \Sigma, w)$ includes a string $w|u$ with $u\neq N$.*
(ii) *The entry $R^*_{ij}(\Sigma, w)$ includes a string $w|u$ with $u\neq N$.*

In brief terms, Lemma 3 indicates that all stable transitions of the machine Σ with the adversarial input character w are characterised by the matrix $R(m, \Sigma, w)$, as long as $m\geq n-1$.

Of course, the adversarial input character is usually not known. To accommodate this fact, we introduce the following notion.

Definition 4: Let Σ be an asynchronous machine with the adversarial input uncertainty ν. The one-step stable transitions matrix $\rho(\Sigma, \nu)$ of Σ consists of the entries

$$\rho_{ij}(\Sigma, \nu) := \vee_{w\in\nu}\ \rho_{ij}(\Sigma, w), \quad i,j=1,2,\ldots,n.$$

The matrix $\rho(\Sigma, \nu)$ simply includes all one-step stable transitions that are compatible with the adversarial uncertainty of Σ.

4.3 *State feedback*

Recall that our objective is to build a state feedback controller C that turns the closed loop machine Σ_c of Figure 1 into a deterministic machine not affected by the adversarial input. The following notion forms the basis of our forthcoming discussion; compare to Venkatraman and Hammer (2006b,c).

Definition 5: Let $\Sigma=(A\times B, X, f, \nu)$ be an asynchronous machine with an adversarial input, and let $x^i, x^j\in X$ be two states of Σ. We say that there is a *feedback path* from x^i to x^j if there is a state feedback controller that takes Σ from a stable combination with x^i to a stable combination with x^j in fundamental mode, given only that the adversarial input is within the uncertainty set ν.

In these terms, our objective is to find pairs of states of the machine Σ that can be connected by a feedback path. Note that, due to fundamental mode operation, the adversarial input character does not change along a feedback path, but its value is not, in general, known. The following is a simple property of feedback paths.

Lemma 4: *Let $\Sigma=(A\times B, X, f)$ be an asynchronous machine with the matrix of stable transitions $R(m, \Sigma, \nu)$. Let $x^i, x^j\in X$ be two states of Σ, and assume that there is a feedback path from x^i to x^j. If w is a possible adversarial input value along this path, then $w\in\Pi_a R_{ij}(m, \Sigma, \nu)$ for some $m\geq 1$.*

Proof: If there is a feedback path from x^i to x^j with the adversarial input value w, then there is a control input string that takes Σ from x^i to x^j through a string of stable transitions, while the adversarial input character is w. Thus, $w\in\Pi_a R_{ij}(m, \Sigma, \nu)$, and the proof concludes. □

Lemma 4 provides a simple necessary condition for the existence of a feedback path. We derive a sufficient condition for the existence of feedback paths in the next section.

5. Complete sets of strings

Let $\Sigma=(A\times B, X, f, \nu)$ be an asynchronous machine with an adversarial input, and let x^i and x^j be two states of Σ. In this section, we develop a test to determine whether there is a feedback path from x^i to x^j. Critical to this development is the uncertainty about the adversarial input value. This uncertainty may vary along a feedback path due to the fact that the controller accumulates information about the adversarial input value.

As an example, assume that ν consists of two characters, say $\nu=\{w^1, w^2\}$. Then, initially, it is known only that the adversarial input value is one of the characters w^1 or w^2. Now, assume that the control input value is changed to the character u'. Letting s be the stable recursion function of Σ, we have two options for the next stable state:

(i) $x' := s(x, u', w^1)$ when the adversarial input character is w^1; and
(ii) $x'' := s(x, u', w^2)$ when the adversarial input character is w^2.

Clearly, if $x'\neq x''$, then we can determine the value of the adversarial input character from the next stable state, thus resolving the uncertainty. On the other hand, if $x'=x''$, then the outcome of this step does not reduce the uncertainty about the adversarial input.

In summary, the uncertainty about the adversarial input value may be reduced as we progress along a feedback path. Of course, only the uncertainty

changes—the adversarial input value itself is constant along a feedback path in fundamental mode operation.

The adversarial uncertainty affects the selection of the next control input character, as we now discuss. Consider the case where the machine Σ is at a stable combination with the state x and the control input value u_0, while being driven along a feedback path toward a stable combination with the state x'. Let $\nu_0 \subset \nu$ be the current uncertainty about the value of the adversarial input, and let A be the control input alphabet. Let S be the set of all strings that take Σ from its current state x to a stable combination with the state x', i.e., all strings $w|u = w|u_0u_1 \ldots \in \nu_0|A^+$ for which $s(x, u, w) = x'$. Letting u_1 be the next control input character, denote by $S(a)$ the subset of all strings of S for which $u_1 = a$, where $a \in A$ is a character of the control input alphabet. Then, the set of adversarial input characters that are compatible with the control input character a at step 1 is given by $\Pi_a\, S(a)$.

Now, if $\nu_0 \not\subset \Pi_a\, S(a)$, then the character a cannot be used as the control input at step 1, since it is not compatible with some adversarial input values that may presently be active. On the other hand, if $\nu_0 \subset \Pi_a\, S(a)$, then a can be applied as the next control input character, since it is compatible with the information currently available about the adversarial input value. To conclude, the current level of adversarial uncertainty ν_0 impacts the selection of the next control input character.

For a member $\sigma = w|u_0u_1 \ldots u_k$ of S and an integer $q \geq 0$, it is convenient to define the truncated string

$$\sigma|_q := \begin{cases} w|u_0u_1 \ldots u_q & \text{if } q \leq k, \\ w|u_0u_1 \ldots u_k & \text{if } q > k. \end{cases}$$

The set of all truncated members of S is denoted by

$$S|_q := \{\sigma|_q : \sigma \in S\}, \quad q = 1, 2, \ldots.$$

Recall that S includes all strings that take Σ from a stable combination with the state x to a stable combination with the state x'. Then, the string $\sigma|_q$ takes Σ to a stable combination with the state

$$x_q := s(x, \sigma|_q) := s(x, u_0u_1 \ldots u_q, w), \quad q = 1, 2, \ldots.$$

The stable states that Σ passes while being driven by the string σ are given by the list

$$x_0(\sigma) := s(x, \sigma|_0), x_1(\sigma) := s(x, \sigma|_1), \ldots, x_k(\sigma) := s(x, \sigma|_k),$$

where $x_0(\sigma) = x$ and $x_k(\sigma) = x'$.

For a string $\sigma = w|u_0u_1 \ldots u_k \in S$, we define the projection $\Pi^p\colon S \to A$ which extracts the pth control input character of σ, i.e.,

$$\Pi^p\sigma := \begin{cases} u_p & \text{for } p = 0, 1, \ldots, k, \\ u_k & \text{for all } p > k. \end{cases}$$

Next, assume that the machine Σ is operated by a state feedback controller C that uses strings from the set S to drive Σ, while the adversarial input value w is kept constant. As usual, there is no direct information about the value of w. However, the control input values of Σ and the states through which Σ passes are known to the controller, as the controller generates the input values and reads the states of Σ. This data can be used to reduce the uncertainty about the adversarial input value w, as follows.

Let x and x' be two states of Σ, let u be a control input string of Σ, and let s be the stable recursion function of Σ. Assume that Σ is in a stable combination with the state x when the control input value changes to u, and let x'' be the next stable state of Σ. Define the *adversarial inverse function* s^a by setting

$$s^a(x, u, x'') := \{w \in B : s(x, u, w) = x''\}, \tag{8}$$

so that $s^a(x, u, x'')$ is the set of all adversarial input values $w \in \nu$ that are compatible with the stable transition $s(x, u, w) = x''$. In particular, when Σ is at a stable combination with the initial state x_0 and the control input value u_0, it follows from (8) that the adversarial input character w must satisfy

$$w \in \nu(x_0, u_0) := s^a(x_0, u_0, x_0) \cap \nu. \tag{9}$$

Thus, the initial uncertainty about the adversarial input value may, in fact, be smaller than ν.

Recall that S is the set of all strings that take Σ from a stable combination with the state $x_0 := x$ to a stable combination with the state x'. At the initial step, the set of all possible adversarial input values is given by (9). Consequently, the set S must contain a path for each adversarial input character $w \in \nu\,(x_0, u_0)$, namely, we must have $\nu\,(x_0, u_0) \subset \Pi_a\, S$. Otherwise, the set S would be incompatible with some of the potential adversarial input values.

Further, let u_1 be a control input character, and let $S(x_0, u_0u_1)$ be the set of all strings of S whose control input starts with u_0u_1, i.e.,

$$S(x_0, u_0u_1) = \{\sigma \in S : \sigma|_1 = w|u_0u_1 \quad \text{for some } w \in B\}.$$

Clearly, the character u_1 can be used as the next control input only if it is compatible with all possible adversarial input values, i.e., only if

$$\nu(x_0, u_0) \subset \Pi_a S(x_0, u_0u_1).$$

As the control input string is generated by the controller C, the pair (x_0, u_1) must be detectable to facilitate fundamental mode operation of the closed loop machine.

Now, let x_1 be the next stable state of Σ reached with the control input character u_1. The fact that Σ has

reached the state x_1 implies that the adversarial input value w must have been within the set

$$\nu(x_0x_1, u_0u_1) := s^a(x_0, u_1, x_1) \cap \nu(x_0, u_0).$$

Continuing in this way, suppose that we are at step p of the path. Let $u_0u_1 \dots u_p$ be the control input characters applied so far to Σ along this path by the controller, and let $x_0x_1 \dots x_p$ be the string of stable states through which Σ has passed as a result. Let $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p) \subset B$ be the current uncertainty about the adversarial input value. By iterating the earlier step, we obtain the following conclusion.

Lemma 5: *Let w be the adversarial input character of the asynchronous machine Σ and let $p \geq 1$ be an integer. Assume that the control input string $u_0u_1 \dots u_p \in A^+$ drives Σ through the states $x_0x_1 \dots x_p$, where $(x_i, u_i, w), i = 0, 1, 2, \dots, p$, are all stable combinations. Then, $w \in \nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$, where*

$$\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p) := s^a(x_{p-1}, u_p, x_p) \cap \nu(x_0x_1 \dots x_{p-1}, u_0u_1 \dots u_{p-1}).$$

Referring to Lemma 5, it follows from (9) that $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p) \subset \nu$. We call $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$ the *residual adversarial uncertainty* at step p.

Let us return now to our set of strings S that take the machine Σ from the state $x_0 := x$ to the state x'. Denote by $S(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$ the set of all elements $\sigma \in S$ that satisfy the following conditions:

(i) the control input values are $u_0u_1 \dots u_p$; and
(ii) the machine Σ passes through the states $x_0, x_1, \dots, x_p$.

For a control input character $d \in A$, denote by $S(x_0x_1 \dots x_p, u_0u_1 \dots u_pd)$ the set of all strings $\sigma \in S(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$ that have the character d in position $p+1$ of their control input string. Applying Lemma 5 to step p of the machine Σ, it follows that the adversarial input value must be within the set $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$. Also, the set of all adversarial input characters that appear with the next control input character d is $\Pi_a S(x_0x_1 \dots x_p, u_0u_1 \dots u_pd)$. Combining the last two facts, we obtain the following.

Lemma 6: *The character $d \in A$ can be used as the next control input character of the machine Σ only if $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p) \subset \Pi_a\, S(x_0x_1 \dots x_p, u_0u_1 \dots u_pd)$.*

The condition of Lemma 6 is critical to the construction of a feedback controller that automatically takes a machine Σ with an adversarial input from one specified state to another. In fact, we show later that this condition guaranties the existence of such a controller, if it is valid at every stable transition along the way from x_0 to x'. These considerations lead us to the following.

Definition 6: Let $S \subset B|A^+$ be a set of strings taking the asynchronous machine Σ from a stable combination with the state x_0 to a stable combination with the state x'. The set S is *complete* if the following two conditions hold for all integers $p = 0, 1, 2, \dots$ and for every control input character $d \in \Pi^{p+1} S(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$:

(i) $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p) \subset \Pi_a \quad S(x_0x_1 \dots x_p, u_0u_1 \dots u_pd)$, and
(ii) The pair (x_p, d) is detectable with respect to the residual adversarial uncertainty $\nu(x_0x_1 \dots x_p, u_0u_1 \dots u_p)$.

Our next objective is to show that the existence of a complete set of strings is equivalent to the existence of a state feedback controller. Shortly thereafter, we present an algorithm for the derivation of complete sets of strings. First, we show that a complete set of strings can be replaced by a complete set of strings of bounded length. For a set of strings $S \subset B|A^+$, denote by $|S|$ the maximal length of a control input string in S, i.e., the maximal length of a string of the set $\Pi_c S$. For a finite set Z, denote by $\#Z$ the number of elements of Z.

Lemma 7: *Let $\Sigma = (A \times B, X, f, \nu)$ be an asynchronous machine with n states, and let x_0 and x' be two states of Σ. Assume that Σ is in a stable combination at the state x_0 with the control input value u_0. If there is a complete set of strings from x_0 to x', then there also is such a complete set of strings S satisfying $|S| \leq [\#\nu(x_0, u_0)](n-1)$.*

Proof: Consider a string $\sigma \in S$. Let $u = u_0u_1 \dots u_k = \Pi_c\sigma$ be the control input values of this string and let $x_0x_1 \dots x_k$ be the string of stable states through which Σ passes as a result of receiving the control input string u. The residual adversarial uncertainty at the start of the path is $\nu_0 := \nu(x_0, u_0)$. Let ν_i be the residual uncertainty at step i of the path, and note that, by definition, ν_i is a monotone declining function of i, and its minimal value cannot be less than 1. Divide the interval $[0, k]$ into segments of constant residual uncertainty. This results in the set of $m+1$ subintervals $I = \{[0, i_1], [i_1+1, i_2], \dots, [i_m+1, k]\}$, where ν_i is constant over each one of these intervals. Since ν_i is a monotone declining function and its minimum cannot be less than 1, we get $m+1 \leq \#\nu(x_0, u_0)$, or $m \leq \#\nu(x_0, u_0) - 1$.

Now, if any of the subintervals $[i, i'] \in I$ has length $\ell \geq n$, then the string of states $x_ix_{i+1} \dots x_{i'}$ must contain a repeating state, say $x := x_p = x_r$, where $i \leq p < r \leq i + \ell$. Since $\nu_p = \nu_r$ by construction, the control input value u_p can be replaced by the control value u_r without

disturbing the stable combination at step p (recall that the adversarial input value is constant during the entire path). Then, steps $p+1, p+2, \ldots, r$ can be eliminated from the string, resulting in a new segment with the length of $\ell-(r-p)$. This process can be repeated again and again, until the length of the resulting segment is less than n. Applying the same procedure to each one of the segments in I, we obtain a new path of length not exceeding $(m+1)(n-1) \le [\#\nu(x_0, u_0)](n-1)$. As this bound is valid for every segment in I, our proof concludes. □

We have reached the main result of this section.

Theorem 2: *Let $\Sigma = (A \times B, X, f, \nu)$ be an asynchronous machine and let x and x' be two states of Σ. Then, the following two statements are equivalent.*

(i) *There is a state feedback controller C that drives Σ from a stable combination with x to a stable combination with x' in fundamental mode operation.*

(ii) *There is a complete set of strings $S \subset B|A^+$ taking Σ from a stable combination with x to a stable combination with x'.*

Proof: Assume first that (ii) is valid. We build a state feedback controller $F(x, x', v)$ which, upon receiving the input character $v \in A$, generates a string of control input characters that takes Σ from a stable combination with $x_0 := x$ to a stable combination with x' in fundamental mode operation. To this end, assume that Σ is in a stable combination with the state x_0, and pick a control input character $u_1 \in \Pi^1 S$. Due to the fact that S is a complete set of strings, the pair (x_0, u_1) is detectable with respect to the adversarial uncertainty $\nu(x_0, u_0)$. In addition, $\nu(x_0, u_0) \subset \Pi_a S(x_0, u_0 u_1)$, so that the input character u_1 is compatible with every possible adversarial input value.

Now, let Ξ be the state set of the controller $F(x, x', v)$. The recursion function ϕ of $F(x, x', v)$ has three variables: the state of $F(x, x', v)$, the state of Σ, and the external control input, i.e., $\phi: \Xi \times X \times A \to \Xi$. Denote by $\eta: \Xi \times X \times A \to A$ the output function of $F(x, x', v)$, and let ξ_0 be the initial state of $F(x, x', v)$. We construct next the functions ϕ and η.

Upon encountering a detectable transition of Σ to the state x_0 with the control input value u_0, the controller moves to a stable combination with the state ξ_1. This transition prepares the controller to generate the input string that will take Σ to the state x', when commanded to do so; it is accomplished by setting

$$\begin{aligned} \phi(\xi_0, (z, t)) &:= \xi_0 \quad \text{for all } (z, t) \neq (x_0, u_0), \\ \phi(\xi_0, (x_0, u_0)) &:= \xi_1, \\ \phi(\xi_1, (x_0, u_0)) &:= \xi_1. \end{aligned}$$

While in its initial state ξ_0 or in the state ξ_1, the controller applies to the control input of Σ the external input character it receives, making $F(x, x', v)$ transparent in these states:

$$\begin{aligned} \eta(\xi_0, (z, t)) &:= t \quad \text{for all } (z, t) \in X \times A, \\ \eta(\xi_1, (z, t)) &:= t \quad \text{for all } (z, t) \in X \times A. \end{aligned}$$

Suppose now that, while $F(x, x', v)$ is in the state ξ_1, it receives the external input character $v \in A$. This is the command for the controller to start a string of transitions taking Σ from its current stable combination with the state x_0 to a stable combination with the state x'. Upon receiving the external input value v, the controller $F(x, x', v)$ moves to a stable combination with the state ξ_2, namely,

$$\begin{aligned} \phi(\xi_1, (z, t)) &:= \xi_1 \quad \text{for all } (z, t) \neq (x_0, v), \\ \phi(\xi_1, (x_0, v)) &:= \xi_2, \\ \phi(\xi_2, (x_0, v)) &:= \xi_2. \end{aligned}$$

When reaching the state ξ_2, the controller applies to the control input of Σ the first character of the control input string $u_1 u_2 \ldots u_k \in \Pi_c S$ that ultimately takes Σ to the state x'; to this end, set

$$\eta(\xi_2, (x_0, t)) := u_1 \quad \text{for all } t \in A.$$

The control input character u_1 causes Σ to move to the state x_1 through a detectable transition.

Continuing in this manner, assume that the controller $F(x, x', v)$ has generated so far the control input string $u_0 u_1 \ldots u_p$, taking Σ through the states $x_0 x_1 \ldots x_p$; here, p is an integer between 1 and k. Since S was a complete set of strings, the transition to the state x_p was a detectable transition; consequently, Σ is in a stable combination when it reaches the state x_p. Upon detecting the state x_p, the controller $F(x, x', v)$ moves to a stable combination with the state ξ_{p+2}, namely,

$$\begin{aligned} \phi(\xi_{p+1}, (z, t)) &:= \xi_{p+1} \quad \text{for all } (z, t) \neq (x_p, \vee), \\ \phi(\xi_{p+1}, (x_p, v)) &:= \xi_{p+2}, \\ \phi(\xi_{p+2}, (x_p, v)) &:= \xi_{p+2}. \end{aligned}$$

Now, select any control input value $u_{p+1} \in \Pi^{p+1} S(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p)$. Due to the fact that S is a complete set of strings, the pair (x_p, u_{p+1}) is detectable with respect to the adversarial uncertainty $\nu(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p)$. Also, $\nu(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p) \subset \Pi_a S(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p u_{p+1})$, so that the input character u_{p+1} is compatible with any adversarial character in $\nu(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p)$. Upon reaching the state ξ_{p+2}, the controller applies to Σ the control input character u_{p+1}, namely,

$$\eta(\xi_{p+2}, (x_p, t)) := u_{p+1} \quad \text{for all } t \in A.$$

This construction is repeated for $p = 1, 2, \ldots$, until the machine Σ reaches the state x'. In view of our construction, the resulting controller $F(x, x', v)$ satisfies condition (i) of the Theorem. Note that, by Lemma 7, the state x' can be reached at a step $k \leq (n-1)[\#v(x_0, u_0)]$, where n is the number of states of the machine Σ.

Conversely, assume that condition (i) is valid. Let $F(x, x', v)$ be a controller which, upon receiving the external input character $v \in A$, takes Σ from a stable combination with the state $x_0 := x$ to a stable combination with the state x'. Assume that Σ is in a stable combination with the state x_0 and the control input value u_0, when the external input changes to the character v. The initial uncertainty about the adversarial input character of Σ is then $v(x_0, u_0)$. Let $S \subset B|A^+$ be the set of all strings the controller $F(x, x', v)$ can generate. To prove that (i) implies (ii), we need to show that S is a complete set of strings.

To show that S is a complete set of strings, consider step $p \geq 0$ of a string of control input characters $u_0 u_1 \ldots u_p$ applied by $F(x, x', v)$ to Σ. Denote by $x_0 x_1 \ldots x_p$ the stable states through which Σ has passed as a result of this string. By Lemma 5, the residual adversarial uncertainty at this point is $v(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p)$. Let $d \in A$ be the next control input character that F generates for Σ. Then, by fundamental mode operation of the closed loop machine, the pair (x_p, d) is detectable with respect to the residual uncertainty $v(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p)$. Also, in view of Lemma 6, we have $v(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p) \subset \Pi_a S(x_0 x_1 \ldots x_p, u_0 u_1 \ldots u_p d)$. As this is true for all $p \geq 0$, the requirements of Definition 6 are met, and S is a complete set of strings. Thus, (i) implies (ii), and our proof concludes. □

In view of Theorem 2, finding a complete set of strings is the critical step in the process of designing a controller for an asynchronous machine with adversarial input. The following algorithm derives such a set of strings.

Algorithm 1: Derivation of a complete set of strings. Let $\Sigma = (A \times B, X, f, v)$ be an asynchronous machine with adversarial input. Let $S \subset B|A^+$ be a set of strings that take Σ from a stable combination with the state x_0 and the control input value u_0 to a stable combination with the state x'. Consider the case where Σ is at step j of an input string from S, having received the control inputs $u_0 u_1 \ldots u_j$ and having passed through the stable states $x_0 x_1 \ldots x_j$. The residual adversarial uncertainty is $v(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$.

Step 0: Set $j := 0$.

Step 1: If $v(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j) \not\subset \Pi_a S(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$, then S does not include a complete set of strings. Set $\Phi := \varnothing$ and terminate the algorithm. Otherwise, continue to Step 2.

Step 2: Let S_1 be the set of all strings $\sigma \in S(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$ for which $\Pi_a \sigma \notin v(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$. If $S_1 \neq \varnothing$, then replace S by the difference set $S \backslash S_1$ and go to Step 0. If $S_1 = \varnothing$, continue to Step 3.

Step 3: Let S_2 be the set of all strings $\sigma \in S(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$ for which the pair $(x_j, \Pi^{j+1}\ \sigma)$ is not detectable with respect to the residual uncertainty $v(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$. If $S_2 \neq \varnothing$, then replace S by the difference set $S \backslash S_2$ and go to Step 0. If $S_2 = \varnothing$, continue to Step 4.

Step 4: Let S_3 be the set of all strings $\sigma \in S(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j)$ for which $v(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u_j) \not\subset \Pi_a S(x_0 x_1 \ldots x_j, u_0 u_1 \ldots u \Pi^{j+1} \sigma)$. If $S_3 \neq \varnothing$, then replace S by the difference set $S \backslash S_3$ and go to Step 0. If $S_3 = \varnothing$, continue to Step 5.

Step 5: Let q be the length of the longest string in S. If $j = q$, then set $\Phi := S$ and terminate the algorithm. Otherwise, replace j by $j + 1$ and go to Step 1.

The outcome of the Algorithm 1 is a set of strings $\Phi \subset B|A^+$. If Φ is not the empty set, then, according to the next statement, it forms a complete set of strings.

Theorem 3: *Let $\Sigma = (A \times B, X, f, v)$ be an asynchronous machine with adversarial input, let $S \subset B|A^+$ be a set of strings all having the same initial control input character, and let Φ be the outcome of Algorithm* 1. *Then,*

(i) *Φ is not empty if and only if S contains a complete set of strings, and*
(ii) *If Φ is not empty, then it forms a complete set of strings included in S.*

Proof: Assume that Φ is not empty. Then, an examination of Step 3 of Algorithm 1 shows that Φ satisfies condition (i) of Definition 6, while an examination of Step 4 of the algorithm shows that Φ satisfies condition (ii) of Definition 6. A slight reflection on the flow of Algorithm 1 leads then to the conclusion that Φ is a subset of S, and that it is not empty if and only if S contains a complete set of strings. □

6. Skeleton matrices

We turn now to the definition of one of the main notions of our present discussion. Let $\Sigma = (A \times B, X, f, v)$ be an input/state asynchronous sequential machine with adversarial input, having the state set $X = \{x^1, \ldots, x^n\}$ with n states. In view of (9), the initial adversarial uncertainty always satisfies

$\nu(x_0, u_0) \subset \nu$, so we always have $\#\nu(x_0, u_0) \leq \#\nu$. Invoking Lemma 7, we conclude that a complete set of strings S for the machine Σ can always be selected so that its length satisfies

$$|S| \leq (n-1)(\#\nu). \tag{10}$$

Recall that, by (7), the i,j entry of the matrix $R(m, \Sigma, w)$ includes all the control input strings that take Σ from a stable combination with the state x^i to a stable combination with the state x^j through a string of m or fewer stable transitions, while the adversarial input character is w. At this point, it is convenient to introduce the matrix

$$R(\Sigma, \nu) := \vee_{w \in \nu} R((n-1)(\#\nu), \Sigma, w). \tag{11}$$

Definition 7: The matrix $R(\Sigma, \nu)$ is the combined matrix of stable transitions of the asynchronous machine Σ with the adversarial uncertainty ν.

Example 5: Continuing with our analysis of the machine Σ of Example 1, let the adversarial uncertainty be $\nu = \{\alpha, \beta\}$, so that $\#\nu = 2$. As we have three states in this case, $n = 3$, and $(n-1)(\#\nu) = 4$.

$$R(\Sigma, \nu) = \rho(\Sigma, \alpha) \vee \rho(\Sigma, \beta) \vee \rho^2(\Sigma, \alpha) \vee \rho^2(\Sigma, \beta) \vee \rho^3(\Sigma, \alpha) \vee \rho^3(\Sigma, \beta) \vee \rho^4(\Sigma, \alpha) \vee \rho^4(\Sigma, \beta)$$

$$= \begin{pmatrix} \{\alpha|a, \alpha|aba, \alpha|baba, \alpha|ba, \alpha|abab, \alpha|ab, \alpha|bab, \alpha|b, \beta|a, \beta|baba, \beta|aba, \beta|ba\} & \{\beta|abab, \beta|bab, \beta|ab, \beta|b\} & \{\alpha|N, \beta|N\} \\ \{\alpha|baba, \alpha|aba, \alpha|abab, \alpha|bab, \alpha|ab, \beta|a, \beta|baba, \beta|ba\} & \{\alpha|b, \beta|abab, \beta|bab, \beta|ab, \beta|b\} & \{\alpha|ba, \alpha|a\} \\ \{\alpha|baba, \alpha|ba, \alpha|aba, \alpha|abab, \alpha|bab, \alpha|b, \alpha|ab, \beta|baba, \beta|ba, \beta|aba\} & \{\beta|abab, \beta|bab, \beta|b, \beta|ab, \beta|a\} & \{\alpha|a, \beta|a\} \end{pmatrix}$$

Considering (10), Lemma 7, and Theorem 2, we reach the following conclusion.

Corollary 1: *Let $\Sigma = (A \times B, X, f, \nu)$ be an asynchronous machine with adversarial input, having the state set $X = \{x^1, \ldots, x^n\}$ and the combined matrix of stable transitions $R(\Sigma, \nu)$. Then, the following two statements are equivalent for all $i, j = 1, 2, \ldots, n$.*

(i) *There is a state feedback controller that takes Σ from a stable combination with x^i to a stable combination with x^j in fundamental mode operation.*
(ii) *The i,j entry of $R(\Sigma, \nu)$ includes a complete set of strings.*

In view of Corollary 1, it is easy to determine whether or not there is a state feedback controller that takes the machine Σ from a stable combination with the state x^i to a stable combination with the state x^j in fundamental mode operation: all we have to do is apply Algorithm 1 to the entry $R_{ij}(\Sigma, \nu)$. Then, such a controller exists if and only if the outcome of Algorithm 1 is a non empty set. This set can then be used to construct an appropriate controller by following the proof of Theorem 2. In this way, we arrive at the following notion.

Definition 8: Let $\Sigma = (A \times B, X, f, \nu)$ be an asynchronous machine with adversarial input, having n states and the combined matrix of stable transitions $R(\Sigma, \nu)$. The complete matrix of stable transitions $\Re(\Sigma, \nu)$ of Σ is an $n \times n$ matrix defined as follows for each $i, j \in \{1, 2 \ldots, n\}$: the entry $\Re_{ij}(\Sigma, \nu)$ is a complete set of strings included in the entry $R_{ij}(\Sigma, \nu)$; if there is no such complete set, then $\Re_{ij}(\Sigma, \nu) := N$.

Example 6: Applying Algorithm 1 on the entries of the matrix of stable transitions derived in Example 5, we obtain

$$\Re(\Sigma, \nu) = \begin{pmatrix} \{\alpha|a, \beta|a\} & N & N \\ \{\alpha|ab, \beta|a, \beta|ba\} & \{\alpha|b, \beta|ab, \beta|b\} & N \\ \{\alpha|ab, \alpha|ba, \alpha|b, \beta|ba\} & N & \{\alpha|a, \beta|a\} \end{pmatrix}.$$

In view of Theorem 2, the following is true.

Corollary 2: *Let Σ be an asynchronous sequential machine with the state set $\{x^1, \ldots, x^n\}$ and the adversarial uncertainty ν. Let $\Re(\Sigma, \nu)$ be the complete matrix of stable transitions of Σ. Then, the following two statements are equivalent for all $i, j \in \{1, \ldots, n\}$:*

(i) *There is a state feedback controller that takes Σ from a stable combination with the state x^i to a stable combination with the state x^j in fundamental mode operation.*
(ii) $\Re_{ij}(\Sigma, \nu) \neq N$.

We can now generalise the notion of the skeleton matrix (Murphy et al. 2002, 2003) to asynchronous machines with adversarial inputs.

Definition 9: Let $\Sigma = (A \times B, X, f, \nu)$ be an input/state asynchronous sequential machine with adversarial input, having the state set $X = \{x^1, \ldots, x^n\}$ and the complete matrix of stable transitions $\Re(\Sigma, \nu)$. The *control skeleton matrix* $K(\Sigma, \nu)$ of Σ is an $n \times n$ matrix of zeros and ones, whose entries are defined as follows for each $i, j \in \{1, 2, \ldots, n\}$: $K_{ij}(\Sigma, \nu) := 1$ if $\Re_{ij}(\Sigma, \nu) \neq N$, and $K_{ij}(\Sigma, \nu) := 0$ if $\Re_{ij}(\Sigma, \nu) = N$.

Example 7: Using the result of Example 6, we obtain the control skeleton matrix

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

In these terms, a state feedback controller can take Σ from a stable combination with the state x^i to a stable combination with the state x^j in fundamental mode operation if and only if $K_{ij}(\Sigma, \nu) = 1$.

6.1 *Latent adversarial switches*

We turn our attention now to a restricted version of the model matching problem with adversarial inputs. The solution of this problem forms a step stone along our way toward the solution of the full model matching problem with adversarial inputs.

Consider an asynchronous machine Σ with adversarial input uncertainty ν. Note that changes in the adversarial input value do not always cause a state transition of Σ. Indeed, assume that Σ is at a stable combination with the state x and the control input character u_0, and consider the set of adversarial input characters $s^a(x, u_0, x)$ of (8). Clearly, if this set consists of more than one character, then a switch of the adversarial input from one character to another in this set does not change the stable state of Σ, and hence is not noticeable by a state feedback controller. This leads to the following.

Definition 10: A *latent adversarial switch* is a change of the adversarial input value that does not result in a change of the stable state of the machine.

The next statement provides a solution of the model matching problem for the case when all adversarial input changes are latent.

Theorem 4: *Let $\Sigma = (A \times B, X, f, \nu)$ be an input/state machine with adversarial input, and assume that the adversarial input is restricted to latent switches. Let $K(\Sigma, \nu)$ be the control skeleton matrix of Σ, and let $\Sigma' = (A, X, s')$ be a stable-state input/state machine with no adversarial input, having the skeleton matrix $K(\Sigma')$. Then, the following two statements are equivalent.*

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

(ii) $K(\Sigma, \nu) \geq K(\Sigma')$.

Example 8: Consider the problem of building a model matching controller for the machine Σ of Example 1 so as to match the model Σ' of Example 2. Using the procedure described in Murphy et al. (2002, 2003), the skeleton matrix of the model Σ' is calculated as

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

The control skeleton matrix $K(\Sigma, \nu)$ was calculated in Example 7. Comparing the two matrices, we obtain that $K(\Sigma, \nu) \geq K(\Sigma')$, so that a model matching controller exists by Theorem 4. We will construct the controller in §9 below.

Proof of Theorem 4: Assume first that (i) is valid, and consider a particular stable transition of the model Σ', say a transition from a stable combination with the state x^i to a stable combination with the state x^j. Then, by definition of the skeleton matrix, $K_{ij}(\Sigma') = 1$. As (i) is valid, the stable state machine $\Sigma_{c|s}$ must also have a transition from a stable combination with the state x^i to a stable combination with x^j. In view of the fact that Σ_c is the machine Σ controlled by the controller C, it follows by Corollary 2 and Definition 9 that $K_{ij}(\Sigma, \nu) = 1$ as well. Thus, $K_{ij}(\Sigma, \nu) = 1$ if $K_{ij}(\Sigma') = 1, i, j = 1, 2, \ldots, n$. Considering that $K(\Sigma, \nu)$ and $K(\Sigma')$ are both matrices of zeros and ones, the latter implies that $K(\Sigma, \nu) \geq K(\Sigma')$, and (i) implies (ii).

Conversely, assume that (ii) is valid. Let $K^1(\Sigma')$ be the one-step skeleton matrix of the model Σ'; see Murphy et al. (2003). Then, $K(\Sigma') \geq K^1(\Sigma')$, and it follows by (ii) that

$$K(\Sigma, \nu) \geq K^1(\Sigma'). \tag{12}$$

Now, let $i, j \in \{1, 2, \ldots, n\}$ be a pair of integers for which $K^1_{ij}(\Sigma') = 1$. Let s' be the stable recursion function of the model Σ', and denote by $V(i, j)$ the set of all control input characters $v \in A$ for which $s'(x^i, v) = x^j$. Note that, for all i,

$$V(i, j) \cap V(i, j') = \varnothing \quad \text{if } j \neq j', \tag{13}$$

since different target states require different inputs. As $K^1_{ij}(\Sigma') = 1$, it follows by (12) that also $K_{ij}(\Sigma, \nu) = 1$. By Corollary 2, there is then a complete set of strings from the state x^i to the state x^j. Select a character $v \in V(i, j)$. In view of Theorem 2, there is controller $F(x^i, x^j, v)$ that takes the machine Σ from a stable combination with x^i to a stable combination with x^j; here, the character v activates the controller. Extend the activation of this controller to all characters $v \in V(i, j)$, so that any character $v \in V(i, j)$ can be used to start the (same) controller action. Denote the resulting controller by $F(x^i, x^j, V(i, j))$.

Next, we define the following operation of *join* for combining two controllers (see also Venkatraman and

Hammer (2006c)). Given two controllers $F(x^i, x^j, V(i,j))$ and $F(x^{i'}, x^{j'}, V(i',j'))$, the join

$$C := F(x^i, x^j, V(i,j)) \vee F(x^{i'}, x^{j'}, V(i',j'))$$

is constructed as follows

(i) When $x^i = x^{i'}$ and the external input character is v, then

$$C := \begin{cases} F(x^i, x^j, V(i,j)) & \text{if } v \in V(i,j), \\ F(x^{i'}, x^{j'}, V(i',j')) & \text{if } v \in V(i',j'). \end{cases}$$

(ii) If $(x^i, x^j) \neq (x^{i'}, x^{j'})$, let the machine Σ be in a stable combination with the state x^i, when the controller C receives the input character v. Then,

$$C := F(x^i, x^j, V(i,j)).$$

This construction of C is consistent by (13); see Venkatraman and Hammer (2006c) for more details.

Now, let $T \subset \{1,2,\ldots,n\} \times \{1,2,\ldots,n\}$ be the set of all pairs of integers for which $K^1_{ij}(\Sigma') = 1$. Then, a slight reflection shows that the joined controller

$$F := \vee_{i,j \in T} F(x^i, x^j, V(i,j))$$

makes the machine Σ match the model Σ'. By Theorem 2, the closed loop machine is well defined and operates in fundamental mode. This concludes our proof. □

7. General model matching

7.1 *Adversarial detectability*

Let $\Sigma = (A \times B, X, f, v)$ be an asynchronous machine with adversarial input. Assume that Σ is at a stable combination (x, u, w), when the adversarial input value changes to w'. This change may or may not cause Σ to experience a transition to a new stable state. Presently, consider the case when the change in the adversarial input from w to w' causes Σ to move to a new state $x' \neq x$. We refer to such a transition as an *adversarial transition*. In this section, we discuss the existence and the design of a state feedback controller C^a that automatically counteracts adversarial transitions of Σ.

A basic requirement is, of course, that the controller C^a operate in fundamental mode to guarantee deterministic behaviour of the closed loop machine. Being a state feedback controller, C^a has access to the current state of Σ, and it generates the control input of Σ. To obtain fundamental mode operation of the closed loop machine with the controller C^a, it must be possible to determine from the state of Σ whether or not Σ has reached its next stable combination. This leads us to the following, which is closely analogous to Definition 2.

Definition 11: Let $\Sigma = (A \times B, X, f)$ be an input/state asynchronous machine with adversarial uncertainty ε. Assume that Σ is in a stable combination at the state x with the control input character u, when a change in the adversarial input causes Σ to move to the state x'. Then, the pair (x, u) is adversarially detectable with respect to the adversarial uncertainty ε if it can be determined from the current state of Σ whether or not Σ has reached its next stable combination.

Without adversarial detectability, it is not possible to guarantee fundamental mode operation of a closed loop machine controlled by a state feedback controller. In other words, operation must be restricted to adversarially detectable pairs of the controlled machine Σ.

Assume then that the machine Σ is at a stable combination (x, u, w), when the adversarial input character changes to w', causing Σ to transition to a stable combination with the state $x' \neq x$. This transition may, of course, consist of a number of intermediate steps, say $x_0 := x, x_1 := f(x_0, u, w')$, $x_2 = f(x_1, u, w'), \ldots, x_q := f(x_{q-1}, u, w') = x', x_q = f(x_q, u, w')$. Similarly to (5) and (6), we denote

$$\begin{cases} \theta(x, u, w') := x_1 \ldots x_q, \\ \theta[x, u, \varepsilon] := \{\theta(x, u, w') : w' \in \varepsilon\}. \end{cases} \tag{14}$$

The following statement is closely analogous to Theorem 1 and has a similar proof.

Theorem 5: *Let $\Sigma = (A \times B, X, f, v)$ be an input/state asynchronous machine with adversarial input. Assume that Σ is in a stable combination with the state x and the control input value u. In the notation of* (4) *and* (14), *the following two statements are equivalent.*

(i) *The pair (x, u) is adversarially detectable with respect to the adversarial uncertainty v.*
(ii) *States of the set $s^v(x, u)$ appear only at the end of strings belonging to $\theta[x, u, v]$.*

As indicated earlier, to guarantee fundamental mode operation of the closed loop machine, the use of the machine Σ must be restricted to adversarially detectable pairs. This leads us to the following notion. (For a string $\sigma = w|u_1 u_2 \ldots u_q \in B \times A^+$, denote by $\Pi_c^+ \sigma := u_q$ the last control input character of the string.)

Definition 12: Let $\Sigma = (A \times B, X, f, v)$ be an asynchronous machine having adversarial uncertainty v, n states, and the combined matrix of stable transitions $R(\Sigma, v)$. The reduced matrix of stable transitions $R^r(\Sigma, v)$ of Σ is obtained by removing from each column $j = 1, 2, \ldots, n$ of $R(\Sigma, v)$ all strings σ for which the pair $(x^j, \Pi_c^+ \sigma)$ is not adversarially detectable with respect to the uncertainty v.

Example 9: We calculate now the reduced matrix of stable transitions for the matrix $R(\Sigma, \nu)$ of Example 5. Considering the transition table of the machine Σ as provided in Example 1, note that there is only one transition that can be caused by the adversarial input, namely, the transition initiated by a switch of the adversarial input from the character α to the character β, while Σ is at the state x^1 and the control input is b. Symbolically, the transition can be represented by $(x^1, b, \alpha) \to (x^1, b, \beta) \to (x^2, b, \beta)$. For this transition, recalling that s is the stable recursion function of Σ, we have $s(x^1, b, \beta) = x^2$. Hence, $\theta[x^1, b, \alpha] = x^1$ and $\theta[x^1, b, \beta] = x^2$, so that

$$\theta[x^1, b, \nu] = \{x^1, x^2\}.$$

In this case, we have

$$s^{\nu}(x^1, b) = \{x^1, x^2\}.$$

As we can see, the states x^1, x^2 appear only at the end of strings belonging to $\theta[x^1, b, \nu]$. Therefore, by Theorem 5, the pair (x^1, b) is adversarially detectable, and we have $R^r(\Sigma, \nu) = R(\Sigma, \nu)$ in this case.

The reduced matrix of stable transitions characterises all transitions of the machine Σ that end at adversarially detectable pairs. This matrix forms the basis for designing controllers that can counteract adversarial transitions.

7.2 *Reversing adversarial transitions*

Assume that the machine Σ is at an adversarially detectable stable combination with the pair $(x^s, u) \in X \times A$, when a change in the adversarial input causes Σ to move to a stable combination with the state x^t; of course, by fundamental mode operation, the control input value u is kept constant during this process. As this transition started from an adversarially detectable pair, a state feedback controller can determine from the current state of Σ whether or not Σ has reached its next stable combination. Having been caused by the adversarial input, this transition is undesirable; our objective is to design a state feedback controller C^a that automatically reverses this transition. In this way, C^a will counteract the effects of the adversarial input.

Consider then an adversarial transition from a stable combination with the pair (x^s, u) to a stable combination with the pair (x^t, u). Letting ν be the adversarial uncertainty, the set of adversarial input characters that can give rise to such a transition is

$$\nu(x^s, x^t, u) := s^a(x^s, u, x^t) \cap \nu. \tag{15}$$

Clearly, this transition is possible if and only if $\nu(x^s, x^t, u) \neq \varnothing$, and we reach the following.

Definition 13: Let Σ be an asynchronous machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the adversarial uncertainty ν, and assume that Σ is in a stable combination with the control input character u. Then, for a pair of integers $s, t \in \{1, 2, \ldots, n\}$, the adversarial transition indicator is

$$K(x^s, x^t, u) := \begin{cases} 1 & \text{if } \nu(x^s, x^t, u) \neq \varnothing, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The discussion leading to Definition 13 implies the next statement.

Lemma 8: *Assume that the closed loop machine Σ_c is in a stable combination at the state x^s with the control input character u. Then, the following two statements are equivalent.*

(i) *There is an adversarial transition to a stable combination with the state x^t.*
(ii) $K(x^s, x^t, u) = 1$.

To address the question of whether an adversarial transition is reversible or not, it is convenient to introduce the following.

Definition 14: Let Σ be an asynchronous machine with adversarial input. An adversarial transition from a stable combination with the state x^s to a stable combination with the state x^t is reversible if there is a state feedback controller that drives Σ back to a stable combination with the state x^s, without specific information about the adversarial character that caused the transition.

To examine reversible transitions, assume that the machine Σ is in a stable combination with the state x^s and the control input character u, when an adversarial transition to the state x^t occurs. In view of (15), the adversarial uncertainty immediately after the transition is $\nu(x^s, x^t, u)$. Using the reduced matrix of stable transitions $R^r(\Sigma, \nu)$, we construct the scalar function

$$K^r(x^s, x^t, u) := \begin{cases} 1 & \text{if } R^r_{ts}(\Sigma, \nu) \text{ includes a complete set of strings with respect to the adversarial uncertainty } \nu(x^s, x^t, u), \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

A slight reflection indicates that the following is true.

Lemma 9: *In the above notation, an adversarial transition from x^s to x^t is reversible if and only if $K^r(x^s, x^t, u) = 1$.*

Next, let $U(x^s) \subset A$ be the set of all control input characters of the machine Σ that may appear in stable

combinations with the state x^s. For various practical considerations, the designer of the state feedback controller C may choose to avoid using some of these control input characters. The set of active control input characters of Σ at the state x^s is the subset $S(x^s) \subset U(x^s)$ of all control input characters that may be applied by the controller C, while the closed loop machine Σ_c is in a stable combination at the state x^s of Σ. If x^s does not appear as part of a stable state of the closed loop machine, then $S(x^s) := \varnothing$. When all states of Σ appear as stable states of the closed loop machine Σ_c and all possible control input characters of Σ are utilised, we have

$$S(x^s) = U(x^s), \quad s = 1, 2, \ldots, n. \tag{18}$$

Equality (18) describes the most common situation. When it is valid, we say that there are no idle control input characters.

Definition 15: Let Σ be an asynchronous machine with adversarial input having the n states $x^1, \ldots, x^n$. Let $S(x^s)$ be the set of active control input characters at the state x^s. The reversal matrix $A(S, \Sigma)$ is an $n \times n$ numerical matrix with the entries

$$A_{st}(S, \Sigma) := \begin{cases} \min\{K^r(x^s, x^t, u) - K(x^s, x^t, u) : u \in S(x^s)\} & \text{if } S(x^s) \neq \varnothing, \\ 0 & \text{if } S(x^s) = \varnothing, \end{cases}$$

$s, t = 1, 2, \ldots, n$.

For a numerical matrix D, the inequality $D \geq 0$ means that D has no negative entries.

Example 10: We continue our examination of the machine Σ of Example 9. In view of Example 9, there is only one adversarial transition we have to consider, namely, the transition $x^2 = s(x^1, b, \beta)$. Assume that the controller is implemented with no idle control input characters, so that $S(x^1) = U(x^1)$. From (15), we have

$$\nu(x^1, x^2, b) = s^a(x^1, b, x^2) \cap \nu = \{\beta\} \cap \{a, \beta\} = \{\beta\}.$$

From (16), we get that $K(x^1, x^2, b) = 1$. In Example 9, we have seen that $R^r(\Sigma, \nu) = R(\Sigma, \nu)$. Also, from Example 6, we have the complete set of strings $\mathfrak{R}_{21}(\Sigma, \nu) = \{\alpha|ab, \beta|a, \beta|ba\}$. Consequently, $R^r_{21}(\Sigma, \nu)$ includes a complete set of strings with respect to the adversarial uncertainty $\nu(x^1, x^2, b) = \{\beta\}$. Substituting into (17), we obtain that $K^r(x^1, x^2, b) = 1$, so that $A_{12}(U, \Sigma) = 0$. As there are no adversarial transitions other than the transition from x^1 to x^2, we have that $K(x^s, x^t, u) = 0$ for all $(s, t) \neq (1, 2)$. Whence, $A(U, \Sigma) \geq 0$ in this case.

We can characterise now the conditions under which a transition caused by the adversarial input can be counteracted by a state feedback controller.

Theorem 6: *Let Σ be an asynchronous machine with adversarial input, and let $X = \{x^1, \ldots, x^n\}$ be the state set of Σ. Assume that Σ is operated by a state feedback controller using the active control input character sets $S(x^s), s = 1, 2, \ldots, n$, and let $A(S, \Sigma)$ be the corresponding reversal matrix. Then, the following two statements are equivalent.*

(i) *All adversarial transitions of the closed loop machine can be automatically reversed in fundamental mode operation.*
(ii) $A(S, \Sigma) \geq 0$.

Proof: Consider an adversarial transition from a stable combination with the state x^s to a stable combination with the state x^t. We have two cases here, depending on the set $S(x^s)$ of active control input characters.

Case 1: $S(x^s) = \varnothing$: then, the state x^s does not appear in a stable combination of the closed loop machine, and hence no adversarial transitions can start at the state x^s. Then, $A_{st}(S, \Sigma) = 0$ by Definition 15.

Case 2: $S(x^s) \neq \varnothing$: By Lemma 8, an adversarial transition from x^s to x^t is possible if and only if $K(x^s, x^t, u) = 1$ for a control input value $u \in S(x^s)$. By Lemma 9, this transition can be reversed if and only if $K^r(x^s, x^t, u) = 1$. As $K(x^s, x^t, u)$ and $K^r(x^s, x^t, u)$ can only take the values 0 or 1, we conclude that the adversarial transition from x^s to x^t is reversible if and only if $K^r(x^s, x^t, u) - K(x^s, x^t, u) \geq 0$. Since this is true for all states x^s and x^t and for all input values $u \in S(x^s)$, it follows that (i) and (ii) are equivalent. This concludes our proof. □

Most often, a state feedback controller employs every control input character of the set $U(x^s)$. In such case, the combination of Theorems 4 and 6 yields the following statement, which is the main result of this section.

Theorem 7: *Let $\Sigma = (A \times B, X, f, \nu)$ be an input/state machine with adversarial input, and assume that Σ has no idle control input characters. Let $U(x)$ be the set of control input characters that appear in stable combinations with the state x, let $A(U, \Sigma)$ be the corresponding reversal matrix, and let $K(\Sigma, \nu)$ be the control skeleton matrix of Σ. Let $\Sigma' = (A, X, s')$ be a stable-state input/state machine with no adversarial input, having the skeleton matrix $K(\Sigma')$. Then, the following two statements are equivalent*:

(i) *There is a controller C for which $\Sigma_{c|s} = \Sigma'$, where Σ_c is well posed and operates in fundamental mode.*
(ii) $K(\Sigma, \nu) \geq K(\Sigma')$ *and* $A(U, \Sigma) \geq 0$.

Theorem 7 provides a comprehensive solution to the model matching problem for asynchronous machines with adversarial inputs. As we can see, the solution is entirely characterised by two numerical matrix inequalities. As the model machine Σ' has no adversarial input, the controller C, when it exists, counteracts any effects of adversarial activity.

Example 11: Combining the results of Examples 8 and 10, we conclude that condition (ii) of Theorem 7 is valid for the machine Σ of Example 1 and the model Σ' of Example 2. Therefore, Theorem 7 assures us that there is a controller C that controls Σ so that the closed loop machine Σ_c is stably equivalent to the model Σ', thus solving the perturbed model matching problem in this case. The construction of the controller C is described in §9 below.

8. Controller structure

We summarise now the structure of a controller that solves the model matching problem with adversarial inputs. In general terms, the controller consists of two components: a component that achieves model matching and a component that reverses adversarial transitions.

Consider the problem of controlling the machine Σ to match the model Σ'. The construction of the model matching component of the controller is described in the proof of Theorem 4. Regarding adversarial transitions – these, by their nature, occur while the model Σ' remains in a stable combination, since the model has no adversarial input. An adversarial transition can be characterised as follows (see Figure 4): it is a departure from a stable combination of the closed loop Σ_c that is not preceded by a change of the external command input v. We describe next the controller in brief terms.

(i) The state feedback controller F controls the machine Σ to achieve model matching in response to the external command input v. The controller F is built following the procedure described in the proof of Theorem 4.
(ii) The comparator detects stable states of Σ_c that differ from stable states of Σ'. When such a difference is detected, the comparator activates the state feedback controller F to drive Σ back to a stable combination with the correct state.

An example of the construction of the controller C is provided in the next section.

9. Example

In this section, we construct a controller C that solves the model matching problem for the machine Σ of Example 1 and the model Σ' of Example 2; the adversarial uncertainty is $\nu = \{\alpha, \beta\}$. It can be shown that condition (ii) of Theorem 7 is satisfied in this case. Below, we demonstrate the construction of a model matching controller. Recall that s is the stable transition function of Σ and s' is the stable transition function of Σ'. An examination of the transition tables of the machines Σ and Σ' shows that only the following three transitions of Σ are different from corresponding transitions of Σ':

$$s(x^1, b, \beta) = x^2 :: s'(x^1, b) = x^1;$$
$$s(x^2, a, \alpha) = x^3 :: s'(x^2, a) = x^1;$$
$$s(x^3, b, \beta) = x^2 :: s'(x^3, b) = x^1.$$

Using the procedure described in the proof of Theorem 4, we build three state feedback controllers

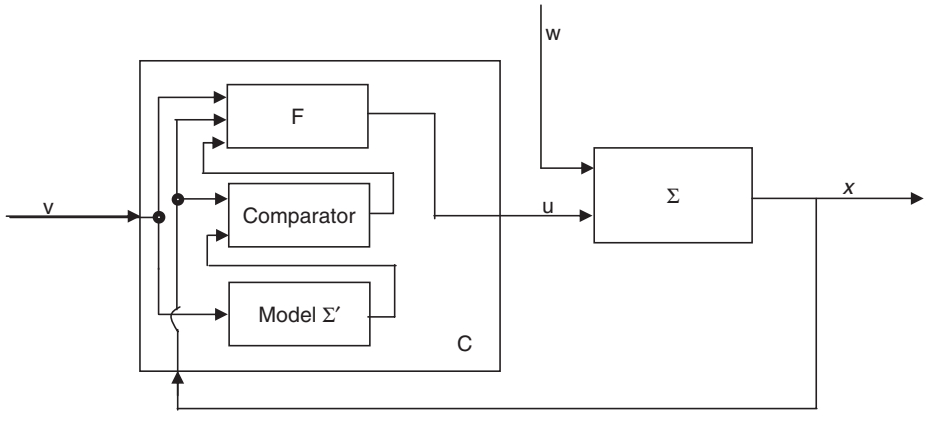

Figure 4. Controller structure.

$F(x^1, x^1, b)$, $F(x^2, x^1, a)$, and $F(x^3, x^1, b)$, each of which respectively 'corrects' one of these transitions.

(i) Construction of the controller $F(x^1, x^1, b)$:

We use the state set $\{\xi^0, \xi^1, \xi^2, \xi^3\}$ for $F(x^1, x^1, b)$. Note that (x^1, a) is a detectable pair. Upon detecting a detectable transition of Σ to the state x^1 with the control input value a, the controller $F(x^1, x^1, b)$ moves to a stable combination with its state ξ^1, while continuing to apply to Σ the input character it receives:

$$\begin{aligned}
\phi(\xi^0, (z, t)) &:= \xi^0 \quad \text{for all } (z, t) \neq (x^1, a),\\
\phi(\xi^0, (x^1, a)) &:= \xi^1,\\
\phi(\xi^1, (x^1, a)) &:= \xi^1.\\
\eta(\xi^0, (z, t)) &:= t \quad \text{for all } (z, t) \in X \times A,\\
\eta(\xi^1, (z, t)) &:= t \quad \text{for all } (z, t) \in X \times A.
\end{aligned}$$

Next, upon receiving the external input character b, the controller $F(x^1, x^1, b)$ moves to a stable combination with its state ξ^2, namely

$$\begin{aligned}
\phi(\xi^1, (z, t)) &:= \xi^1 \quad \text{for all } (z, t) \neq (x^1, b),\\
\phi(\xi^1, (x^1, b)) &:= \xi^2,\\
\phi(\xi^2, (x^1, b)) &:= \xi^2.
\end{aligned}$$

At this point, the controller $F(x^1, x^1, b)$ must start to generate a string of control input characters to keep Σ at the state x^1. An examination of the entry $\Re_{11}(\Sigma, \nu)$ in Example 6 shows that the single control input character a satisfies this requirement. So we set the output function of the controller $F(x^1, x^1, b)$ as

$$\eta(\xi^2, (x^1, t)) := a \quad \text{for all } t \in A.$$

When the controller $F(x^1, x^1, b)$ detects the state x^1 of Σ, it moves to a stable combination with the state ξ^3, namely,

$$\begin{aligned}
\phi(\xi^2, (z, t)) &:= \xi^2 \quad \text{for all } (z, t) \neq (x^1, a),\\
\phi(\xi^2, (x^1, b)) &:= \xi^3,\\
\phi(\xi^3, (x^1, b)) &:= \xi^3,
\end{aligned}$$

and continues to generate the control input character a

$$\eta(\xi^3, (x^1, t)) := a \quad \text{for all } t \in A.$$

Finally, upon a change of the external input character, $F(x^1, x^1, b)$ resets to its initial state ξ^0:

$$\phi(\xi^3, (z, t)) := \xi^0 \quad \text{for all } (z, t) \neq (x^1, b).$$

(ii) The construction of the controllers $F(x^2, x^1, a)$ and $F(x^3, x^1, b)$ is similar to the construction of $F(x^1, x^1, b)$.

(iii) Counteracting adversarial transitions:

Recall from Example 10 that the machine Σ has only one transition that can be caused by the adversarial input – the transition $x^2 = s(x^1, b, \beta)$ when the adversarial input character changes from α to β. Assume then that the machine Σ is at a stable combination with the pair (x^1, b), when the state of Σ switches to x^2. By Example 10, the adversarial uncertainty is then $\nu(x^1, x^2, b) = \{\beta\}$. We build now a controller $F^a(x^2, x^1)$ that counteracts this action of the adversarial input, and returns Σ to the state x^1 immediately after the comparator detects the transition to x^2. The output d of the comparator can take two values: $d := 1$ when the stable state of Σ_c is different from the stable state of Σ', and $d := 0$ when the two stable states are equal.

Referring to Examples 6, 9, and 10, we have that the complete set of strings $\{\alpha|ab, \beta|a, \beta|ba\}$ is included in the entry $\Re_{21}(\Sigma, \nu)$; that $R^r(\Sigma, \nu) = R(\Sigma, \nu)$; and that $\nu(x^1, x^2, b) = \{\beta\}$. Consequently, the single character a forms a control input string that takes Σ back to the state x^1. The construction of the controller $F^a(x^2, x^1)$ is reminiscent of the construction of the controller $F(x^1, x^1, b)$ we have described earlier, and is as follows. Let φ be the recursion function of $F^a(x^2, x^1)$, let μ be its output function, and let $\{\zeta^0, \zeta^1, \zeta^2\}$ be its state set. Starting at a stable combination with the pair (x^2, b), set

$$\begin{aligned}
\varphi(\zeta^0, (z, t, 0)) &:= \zeta^0 \quad \text{for all } (z, t) \neq (x^2, b);\\
\mu(\zeta^0, (z, t)) &:= t \quad \text{for all } (z, t) \in X \times A;\\
\varphi(\zeta^0, (x^2, b, 1)) &:= \zeta^1;\\
\varphi(\zeta^1, (x^2, b, d)) &:= \zeta^1, \quad d \in \{0, 1\};\\
\mu(\zeta^1, (z, t)) &:= a \quad \text{for all } (z, t) \in X \times A;\\
\varphi(\zeta^1, (x^1, b, d)) &:= \zeta^2, \quad d \in \{0, 1\};\\
\varphi(\zeta^2, (x^1, b, d)) &:= \zeta^2, \quad d \in \{0, 1\};\\
\mu(\zeta^2, (z, t)) &:= a \quad \text{for all } (z, t) \in X \times A;\\
\varphi(\zeta^2, (z, t, d)) &:= \zeta^0 \quad \text{for all } (z, t) \neq (x^1, b).
\end{aligned}$$

Finally, we assemble the combined state feedback controller F by using the join operation employed in the proof of Theorem 4:

$$F = F(x^1, x^1, b) \vee F(x^2, x^1, a) \vee F(x^3, x^1, b) \vee F^a(x^2, x^1).$$

When this controller is inserted into Figure 4, it eliminates the effects of the adversarial input and makes Σ behave like the deterministic and unperturbed model Σ' of Example 2.

Acknowledgements

The work of Yang was supported by the Research Grants of the Catholic University of Daegu, while that of Hammer was supported in part by the US Air Force, through grant number FA8750-06-1-0175.

References

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

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

Eilenberg, S. (1974), *Automata, Languages and Machines*, New York: Academic Press.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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