Preprints of the 16th IFAC Symposium on System Identification
The International Federation of Automatic Control
Brussels, Belgium. July 11-13, 2012

Adaptive Control of Asynchronous Sequential Machines: An Algebraic Formulation

Jung-Min Yang * **Tan Xing** ** **Jacob Hammer** **

* *Department of Electrical Eng., Catholic University of Daegu Gyeongsan, Gyeongbuk, 712-702, Korea*
** *Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA. (e-mail: hammer@mst.ufl.edu)*

Abstract: An algebraic framework is developed and utilized to achieve adaptive control of asynchronous sequential machines with unknown transitions. The framework yields adaptive state feedback controllers that acquire data about unknown transitions during normal operation and utilize this data to improve closed-loop performance.

1. INTRODUCTION

Asynchronous sequential machines, or clockless logic circuits, are employed in high speed computing, in parallel computing, in modeling of signaling chains in molecular biology (Hammer [1994, 1995]), and in other applications. We examine the control of asynchronous machines that are not fully described due to incomplete characterization, malfunctions, or errors (e.g., Shieh, Wey, and Fisher [1993]).

An *indeterminate transition* of a machine is a deterministic transition with unknown outcome. Once tested, an indeterminate transition becomes a *determinate* transition – a deterministic transition with known outcome. An *indeterminate machine* is a deterministic machine with indeterminate transitions. We develop algorithms to acquire data about indeterminate transitions without hindering user experience. A controller that collects and uses such data is an *adaptive controller*.

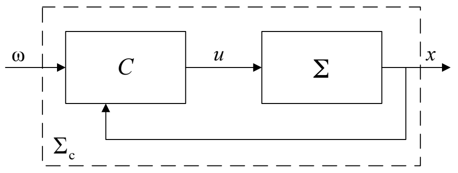

Fig. 1. Closed Loop Configuration

In Figure 1, Σ is an indeterminate asynchronous machine and C is another asynchronous machine serving as an adaptive controller; Σ_c denotes the closed loop.

Asynchronous machines have stable and transient states. A machine lingers at a stable state until an input change occurs; it passes a transient state quickly (ideally, in zero time). Transient states do not affect user experience, but controllers can record transients and use the data to improve performance. In §5, we develop adaptive controllers that deliberately create transients of Σ_c to test indeterminate transitions of Σ and use this data to improve performance. The results are applied to model matching: Given a determinate asynchronous machine model Σ', we design adaptive controllers C satisfying $\Sigma_c = \Sigma'$, where equality refers to stable transitions.

Fundamental mode operation prohibits the simultaneous change of two or more variables. For Figure 1, this means that C and Σ cannot change state simultaneously (Murphy, Geng, and Hammer [2003]). Fundamental mode operation is observed in all our designs; it prevents uncertainties that may arise when simultaneous changes appear instead as sequential changes in unpredictable order.

Our discussion is within the framework of Murphy, Geng, and Hammer [2002, 2003], Geng and Hammer [2005], Venkatraman and Hammer [2006b,c], and Yang and Hammer [2008a,b], where control of asynchronous sequential machines is examined. Other aspects of the control of sequential machines are found in Ramadge and Wonham [1987], Thistle and Wonham [1994], and Kumar, Nelvagal, and Marcus [1997], where discrete event systems are investigated; in Hammer [1994, 1995, 1996a], HAMMER [1996b], Dibenedetto, Saldanha, and Sangiovanni-Vincentelli [1994], Barrett and Lafortune [1998], and Yevtushenko, Villa, Brayton, Petrenko, and Sangiovanni-Vincentelli [2008], where control and model matching of sequential machines is studied; and in other sources.

This note is organized as follows. The algebraic framework of Yang, Xing, and Hammer [2011b] is reviewed in §2. Section 3 explores adaptive model matching, while §4 and §5 develop design algorithms for adaptive controllers.

2. AN ALGEBRA OF INDETERMINACY

An input/state asynchronous machine $\Sigma = (A, X, x^0, f)$ has input alphabet A, state set X, initial state x^0, and recursion function $f : X \times A \to X$ – a partial function. Given an input sequence $u_0u_1u_2\ldots$, it creates a sequence of states $x_{k+1} = f(x_k, u_k)$, $k = 0, 1, 2, \ldots$, where $x_0 := x^0$. A pair $(x, u) \in X \times A$ is *valid* if it is in the domain of f. A valid pair (x, u) is a *stable combination* if $f(x, u) = x$; otherwise, it is a *transient combination*. A machine lingers at a stable combination until an input change occurs; it passes quickly through transient combinations (ideally, in zero time). Applying an input character u at a state x can give rise to a chain of transitions $x_1 := f(x, u)$, $x_2 := f(x_1, u)$, ... If this chain terminates, then $x_i = f(x_i, u)$ for some integer $i \geq 1$, and x_i is

⋆ The work of J.-M. Yang was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2011-0005116).

the *next stable state*. If the chain does not terminate, there is an infinite cycle. This note concentrates exclusively on machines with no infinite cycles.

The *stable recursion function* s of Σ is defined at every valid pair (x,u) by $s(x,u) := x'$, where x' is the next stable state. The stable recursion function describes a user's experience, since transients pass very quickly. Still, transients play an important role, since the controller C of Figure 1 works by turning undesirable stable combinations of Σ into transient combinations of Σ_c ([Murphy, Geng, and Hammer , 2003]). It also uses transients to gauge the response of Σ at indeterminate transitions without impeding user experience.

For an input string $u = u_0u_1 \cdots u_q$, the notation $s(x,u)$ denotes the final stable state reached by Σ, when u is applied starting at the state x. For fundamental mode operation, u must be applied character-by-character, waiting after each character for Σ to reach its next stable state, before applying another character. Denote by A^* the set of all strings of characters of A.

Due to malfunctions, errors, or incomplete characterization, there might be several options for a machine's next state. Regarding f as a set valued function, all possible next states at an indeterminate pair (x,u) form the set $f(x,u)$. The stable recursion function is then set valued as well. To guaranty fundamental mode operation, indeterminate pairs (x,u) at which $x \in s(x,u)$ must not be accessed and are considered invalid (Yang, Xing, and Hammer [2011b,a]).

2.1 *The Adjusted Machine*

Consider an indeterminate machine $\Sigma = \big(A,X,x^0,f\big)$ with the stable recursion function s and the set of indeterminate pairs

$$U := \{(z^1,u^1),(z^2,u^2),...,(z^r,u^r)\} \subseteq X \times A. \tag{1}$$

The set of *potential outcomes* $Z^i := s(z^i,u^i)$, $i = 1,...,r$, is the set of possible next stable states:

$$Z^i := \{z^{i,1},\dots,z^{i,n(i)}\} \subseteq X, n(i) > 1. \tag{2}$$

Formally, we can resolve the indeterminacy by associating a unique 'pseudo' character with each potential outcome: let B be an alphabet disjoint from the input alphabet A; choose a distinct set A^i of $n(i)$ characters of B – one character for each potential outcome

$$\begin{array}{l} A^i := \{u^{i,1},u^{i,2},...,u^{i,n(i)}\} \subseteq B, \text{ where} \\ A^i \cap A^j = \varnothing \text{ for all } i \neq j \in \{1,...,r\}. \end{array} \tag{3}$$

The *pseudo alphabet* is $A' := \cup_{i=1,...r}A^i$, and the *extended input alphabet* is $\tilde{A} := A \cup A'$. Now, build from Σ a determinate machine:

Definition 1. The *adjusted stable recursion function*:

$$s_a(x,u) := \begin{cases} s(x,u) & \text{if } (x,u) \text{ is a determinate pair;} \\ z^{i,j} & \text{if } (x,u) = (z^i,u^{i,j}) \text{ for some} \\ & i \in \{1,...,r\}, j \in \{1,2,...,n(i)\}; \\ z^{i,j} & \text{if } (x,u) = (z^{i,j},u^{i,j}) \text{ for some} \\ & i \in \{1,...,r\}, j \in \{1,2,...,n(i)\}; \\ \text{invalid} & \text{for all other } (x,u) \in X \times \tilde{A}, \\ & \text{including all } (x,u) \in U. \end{cases} \tag{4}$$

Then, $\Sigma_a := (\tilde{A},X,x^0,s_a)$ is the *adjusted machine*. □

In (4), the first line makes Σ_a identical to Σ at determinate pairs of Σ; the second line formally resolves indeterminacy of each indeterminate pair (z^i,u^i) by associating a distinct pseudo input character $u^{i,j} \in \tilde{A}$ with each potential outcome $z^{i,j}$; the third line makes each pair $(z^{i,j},u^{i,j})$ into a stable combination of Σ_a; and the fourth line turns all indeterminate pairs of Σ into invalid pairs of s_a. Then, Σ_a is a determinate 'synthetic' machine, as pseudo characters cannot be applied as real inputs to Σ.

Example 2. Consider an asynchronous machine Σ with state set $X = \{x^1,x^2,x^3,x^4\}$, input alphabet $A = \{a,b,c,d\}$, initial state $x^0 = x^1$, and recursion function described by following table, where the outcomes of the pairs (x^2,c) and (x^3,a) are not fully specified.

state	a	b	c	d
x^1	x^2	x^1	x^1	–
x^2	x^2	–	$\{x^1,x^4\}$	x^3
x^3	$\{x^2,x^4\}$	–	–	x^3
x^4	x^4	x^1	x^4	x^4

Using the pseudo alphabet $A' = \{a^1,a^2,c^1,c^2\}$, the adjusted stable recursion function s_a is given by Table 1. We can see that s_a is a stable recursion function; for instance, $s_a(x^2,c^1) = x^1$ and $s_a(x^1,c^1) = x^1$.

Table 1. The adjusted stable recursion function

state	a	b	c	d	a^1	a^2	c^1	c^2
x^1	x^2	x^1	x^1	–	–	–	x^1	–
x^2	x^2	–	–	x^3	x^2	–	x^1	x^4
x^3	–	–	–	x^3	x^2	x^4	–	–
x^4	x^4	x^1	x^4	x^4	–	x^4	–	x^4

2.2 *Complete Sets and Semirings*

On the set $(\tilde{A})^*$ of all extended alphabet strings, we induce a semiring $\mathfrak{A}$: the product of strings $a,b \in (\tilde{A})^*$ is their concatenation ab; the sum of subsets $c,d \subseteq (\tilde{A})^*$ is the union $c+d := c \cup d$; and the distributive laws are

$$a(b+c) = ab+ac, (a+b)c = ac+bc.$$

Now, consider an indeterminate pair (z^i,u^i) of the machine Σ with the outcomes $\{z^{i,1},...,z^{i,n(i)}\}$, and let $A^i = \{u^{i,1},...,u^{i,n(i)}\}$ be the corresponding pseudo input characters. By (4), we have $z^{i,j} = s_a(z^i,u^{i,j})$ and $z^{i,j} = s_a(z^{i,j},u^{i,j})$. Assume that Σ has a state z' such that, for each outcome $z^{i,j}$, there is an input string $\alpha^j \in A^*$ (with no pseudo characters) satisfying $s(z^{i,j},\alpha^j) = z', j = 1,...,n(i)$. Then, a state feedback controller can always take Σ from (z^i,u^i) to z': detecting $z^{i,j}$, the controller applies to Σ the input string α^j. Thus, the set of strings

$$\gamma^1(i) := \{\ u^{i,1}\alpha^1 + u^{i,2}\alpha^2 + \cdots + u^{i,n(i)}\alpha^{n(i)}\ \}, \tag{5}$$

where $\alpha^1,\alpha^2,...,\alpha^{n(i)} \in A^*$ satisfy $s_a(z^i,u^{i,1}\alpha^1) = s_a(z^i,u^{i,2}\alpha^2) = \cdots = s_a(z^i,u^{i,n(i)}\alpha^{n(i)})$, induces a a determinate stable transition from (z^i,u^i) to a common final stable state.

If Σ reached (z^i,u^i) from a determinate pair (z,u) by an input string $\alpha \in A^*$, then the set of strings

$$\Gamma^1(i) := \{\alpha\gamma^1(i)\} \tag{6}$$

has the features: *a)* it includes a response to every outcome of the indeterminate pair (z^i,u^i); and *b)* each string of the set takes Σ to the same stable state.

Applying this to all indeterminate pairs $\{(z^1,u^1),...,\ (z^r,u^r)\}$ of Σ yields the family of *complete sets of order* 1

$$\Gamma^1(\Sigma) := \cup_{i=1,...,r}\Gamma^1(i); \Gamma^0(\Sigma) := \{\alpha \in A^*\}.$$

Building up, after creating the family $\Gamma^p(\Sigma)$ of complete sets of order p for an integer $p \geq 1$, define

$$\gamma^{p+1}(i) := \{\, u^{i,1}\alpha^1 + u^{i,2}\alpha^2 + \cdots + u^{i,n(i)}\alpha^{n(i)} \,\}, \qquad (7)$$

where $\alpha^1, \alpha^2, ..., \alpha^{n(i)} \in \left[\left(\cup_{j=0,...,p}\Gamma^j(\Sigma)\right)\right]^*$ and $s_a(z^i, u^{i,1}\alpha^1) = s_a(z^i, u^{i,2}\alpha^2) = \cdots = s_a(z^i, u^{i,n(i)}\alpha^{n(i)})$. Then, the family $\Gamma^{p+1}(i)$ of all complete sets of order $p+1$ for the indeterminate pair (z^i, u^i) is

$$\Gamma^{p+1}(i) := \{\alpha\gamma^{p+1}(i)\}, \qquad (8)$$

where $\alpha \in \left[\left(\cup_{j=0,...,p}\Gamma^j(\Sigma)\right)\right]^*$ takes Σ from a determinate pair to the indeterminate pair (z^i, u^i), and the family of all complete sets of order $p+1$ of Σ is

$$\Gamma^{p+1}(\Sigma) := \cup_{i=1,2,...r}\Gamma^{p+1}(i).$$

Definition 3. A *complete set* of Σ is any member of the family $\Gamma(\Sigma) := \cup_{p=0,1,2,...}\Gamma^p(\Sigma)$. □

A complete set induces a determinate transition that may include indeterminate segments. Complete sets are related to state feedback (Yang, Xing, and Hammer [2011b]):

Theorem 4. Let Σ be an indeterminate machine with the adjusted machine $\Sigma_a = (\tilde{A}, X, x^0, s_a)$, and let x and x' be two states of Σ. Then, the following are equivalent:
(i) There is a state feedback controller C that takes Σ through a determinate transition from x to x' in fundamental mode.
(ii) There is a complete set γ satisfying $s_a(x, \gamma) = x'$. □

2.3 Reducing Sets of Input Sequences

Considering that, by Theorem 4, only complete sets matter for the existence of state feedback controllers, we can simplify expressions in the semiring $\mathfrak{A}$ by the following rule

$$\gamma + \alpha = \gamma \text{ for a complete set } \gamma \text{ and a set } \alpha \subseteq (\tilde{A})^*. \qquad (9)$$

A slight reflection shows that the following is also true.

Proposition 5. If γ and γ' are complete sets, then so is $\gamma\gamma'$. □

Definition 6. A set of input strings $S \subseteq (\tilde{A})^+$ is *reducible* if it can be simplified into a complete set by using (9) and Proposition 5; otherwise, S is *irreducible*. A reducible set is in *reduced form* when it is expressed as a complete set. □

By Theorem 4, reducible sets characterize transitions that can be implemented in determinate form by state feedback controllers. Thus, the existence of such state feedback controllers can be deduced by simple algebraic manipulations within the semiring $\mathfrak{A}$.

2.4 Stable Reachability

Applying the matrix of stable transitions of[Murphy, Geng, and Hammer , 2003] to the adjusted machine Σ_a, define

Definition 7. The *one-step adjusted matrix of stable transitions* of Σ has the entries

$$R^a_{pq}(\Sigma, 1) := \begin{cases} \{u \in \tilde{A} \mid s_a(x^p, u) = x^q\} & \text{if not the empty set,} \\ N & \text{else,} \end{cases}$$

where N is a character not in $\tilde{A}$ and $p, q = 1, ..., n$. □

Extend the semiring $\mathfrak{A}$ to include the character N:

$$N\alpha = \alpha N = N, N + \alpha = \alpha + N = \alpha \text{ for all } \alpha \in (\tilde{A})^+.$$

The power $(R^a(\Sigma,1))^i, i = 1, 2, ...,$ is given by the usual definition of matrix multiplication, using the operations of $\mathfrak{A}$. Then, letting n be the number of states of Σ, the combination

$$R^a(\Sigma) := R^a(\Sigma,1) + (R^a(\Sigma,1))^2 + \cdots + (R^a(\Sigma,1))^{n-1} \qquad (10)$$

is called the *adjusted matrix of stable transitions* (Yang, Xing, and Hammer [2011b]).

Definition 8. The *reduced matrix of stable transitions* $R(\Sigma)$ is obtained by writing all reducible entries of $R^a(\Sigma)$ in reduced form; irreducible entries are left unchanged. □

By Murphy, Geng, and Hammer [2003, Lemma 3.9], the matrix $R^a(\Sigma)$ characterizes all possible transitions of the adjusted machine Σ_a. Combining this with Theorem 4, we obtain ([Yang, Xing, and Hammer , 2011b]):

Corollary 9. Let Σ be an asynchronous machine with the state set $\{x^1, x^2, ..., x^n\}$ and the reduced matrix of stable transitions $R(\Sigma)$. Then, the following are equivalent.
(i) There is a state feedback controller that takes Σ through a determinate transition from a stable combination with x^i to a stable combination with x^j in fundamental mode operation.
(ii) $R_{ij}(\Sigma)$ is a complete set. □

The information of the reduced matrix of stable transitions can be condensed ([Murphy, Geng, and Hammer , 2003]):

Definition 10. Let Σ be an asynchronous machine with n states and reduced matrix of stable transitions $R(\Sigma)$. Let Δ be a character not in $\tilde{A} \cup \{N\}$. Then, the *skeleton matrix* $K(\Sigma)$ is an $n \times n$ matrix with the entries

$$K_{ij}(\Sigma) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma) \text{ is a complete set,} \\ 0 & \text{if } R_{ij}(\Sigma) = N, \\ \Delta & \text{if } R_{ij}(\Sigma) \text{ is irreducible.} \end{cases} \qquad □$$

In $K(\Sigma)$, transitions indicated by 1 can be implemented in determinate form by a state feedback controller operating in fundamental mode; transitions indicated by 0 are impossible; and transitions indicated by Δ are indeterminate – these may or may not be possible, depending on outcomes of indeterminate transitions along the way.

3. MODEL MATCHING

Considering Δ as a number

$$0 < \Delta < 1, \qquad (11)$$

the following characterization of model matching was derived in Yang, Xing, and Hammer [2011a,b].

Theorem 11. Let Σ and Σ' be input/state asynchronous machines with the same state set, where Σ' is determinate. Then, the following are equivalent:
(i) There is a state feedback controller C satisfying $\Sigma_c = \Sigma'$, where Σ_c operates in fundamental mode.
(ii) $K(\Sigma) \geq K(\Sigma')$.

Inequality *(ii)* of Theorem 11 fails if *a)* $K(\Sigma)$ has an entry of 0 opposite an entry of 1 in $K(\Sigma')$; or if *b)* $K(\Sigma)$ has an entry of Δ opposite an entry of 1 in $K(\Sigma')$. In *a)*, model matching is impossible; in *b)*, however, model matching is possible if the outcome of certain indeterminate transitions is favorable. This leads us to pre-testing of indeterminate transitions.

4. ADAPTATION

4.1 Transitions to be Tested

By Theorem 11*(ii)*, only transitions involving entries Δ of $K(\Sigma)$ that appear opposite entries of 1 in $K(\Sigma')$ need testing to verify whether model matching is possible. Define the set of pairs

$$D(\Sigma', \Sigma) = \left\{(i,j) \,\middle|\, K_{ij}(\Sigma) = \Delta \text{ and } K_{ij}(\Sigma') = 1\right\}. \qquad (12)$$

Example 12. Let $K(\Sigma)$ be the skeleton matrix of the machine Σ of Example 2, and consider a determinate model Σ' with initial state $x^0 = x^1$ and skeleton matrix $K(\Sigma')$, where

$$K(\Sigma) = \begin{pmatrix} 1 & 1 & 1 & \Delta \\ 1 & 1 & 1 & \Delta \\ 1 & 1 & 1 & \Delta \\ 1 & 1 & 1 & 1 \end{pmatrix}, K(\Sigma') = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \qquad (13)$$

Then, $D(\Sigma', \Sigma) = \{(1,4), (2,4)\}$. □

Let $P(A')$ be the family of all subsets of the pseudo alphabet A'. For the extended alphabet $\tilde{A}$, define the projection $\Pi' : (\tilde{A})^+ \to P(A')$ that extracts all pseudo characters from a string $t \in (\tilde{A})^+$:

$$\Pi' t := \{v \in A' \mid v \text{ is a character of } t\}.$$

For a list of strings $t_1, t_2, ..., t_q \in (\tilde{A})^+$, the projection Π' creates a list of sets, omitting duplicates:

$$\Pi'\{t_1 + t_2 + \cdots + t_q\} := \{\Pi' t_1, \Pi' t_2, ..., \Pi' t_q\}.$$

Example 13. Referring to Example 2, we have

$$\Pi'\{bac^2 + aa^1c^2 + ac^2 + ac^2a^2\} = \{\{c^2\}, \{a^1, c^2\}, \{a^2, c^2\}\}.$$

Then, pseudo characters that are candidates for testing are characterized by:

Definition 14. Let Σ be an asynchronous machine with the reduced matrix of stable transitions $R(\Sigma)$, and let Σ' be a model. The *comparison matrix* $S(\Sigma', \Sigma)$ has the entries

$$S_{ij}(\Sigma', \Sigma) := \begin{cases} \Pi' R_{ij}(\Sigma) & \text{if } (i,j) \in D(\Sigma', \Sigma), \\ \varnothing & \text{otherwise.} \end{cases} \qquad \square$$

An entry of the comparison matrix is a family of sets of pseudo characters. Each member of the (i, j) entry consists of the pseudo characters that appear in one string of $R_{ij}(\Sigma)$, namely, of pseudo characters that facilitate a stable transition from the state x^i to the state x^j. Testing these transitions determines model matching possibilities.

Example 15. Based on Examples 2 and 12:

$$S(\Sigma', \Sigma) = \begin{pmatrix} \varnothing & \varnothing & \varnothing & \left\{ \begin{array}{c} \{c^2\},\{a^1,c^2\},\{a^2,c^2\},\{a^2\}, \\ \{a^1,a^2,c^2\},\{a^1,a^2\},\{c^1,c^2\} \end{array} \right\} \\ \varnothing & \varnothing & \varnothing & \left\{ \begin{array}{c} \{c^2\},\{a^1,c^2\},\{a^2,c^2\},\{a^2\}, \\ \{a^1,a^2,c^2\},\{a^1,a^2\},\{c^1,c^2\} \end{array} \right\} \\ \varnothing & \varnothing & \varnothing & \varnothing \\ \varnothing & \varnothing & \varnothing & \varnothing \end{pmatrix}. \quad \square$$

By Definition 1, a pseudo character $v \in A'$ is associated with an original input character $u(v) \in A$ and with two states $z(v), z'(v)$ of Σ satisfying $z'(v) = s_a(z(v), v) \in s(z(v), u(v))$, i.e., $z'(v)$ is one possible outcome of $(z(v), u(v))$. If $z'' \neq z'(v)$ is another possible outcome of $(z(v), u(v))$, then it is associated with a different pseudo character $w \neq v$, where $z(w) = z(v)$, $u(w) = u(v)$, $z'(w) = z''$, and $z'(w) = s_a(z(w), w) \in s(z(w), u(w))$. Each pseudo character is associated with one distinct outcome.

Now, test an indeterminate pair $(z(v), u(v))$, and let z'_{true} be the outcome. Being a deterministic machine, Σ will always respond with z'_{true} to $(z(v), u(v))$. Define the *true stable recursion function* $s_{true} : X \times A \to X$ by the test outcome, namely,

$$s_{true}(z(v), u(v)) := z'_{true}.$$

A non-empty entry $S_{ij}(\Sigma', \Sigma)$ consists of subsets of pseudo characters; each such subset includes all pseudo characters of an input string taking Σ_a from a stable combination with x^i to a stable combination with x^j. If s_{true} is compatible with all transitions induced by the pseudo characters of one of these subsets, then the transition from x^i to x^j is achieved by (the current sample of) Σ. This leads to

Proposition 16. Let Σ and Σ' be input/state asynchronous machines with comparison matrix $S(\Sigma', \Sigma)$ and identical state sets, where Σ' is determinate. Let s_a be the adjusted stable transition function of Σ and let s_{true} be its true stable recursion function. For a pseudo character $v \in A'$, let $(z(v), u(v))$ be the indeterminate pair associated with v. Then, when $K_{ij}(\Sigma) = \Delta$, the following are equivalent.
(i) $K_{ij}(\Sigma)$ turns into 1 after testing.
(ii) There is a member $\theta \in S_{ij}(\Sigma', \Sigma)$ such that $s_{true}(z(v), u(v)) = s_a(z(v), v)$ for all $v \in \theta$.

Proposition 16 is the basis of our adaptation algorithm.

5. TESTING AND ADAPTATION

5.1 Testing in the Initial State

Testing in the initial state is performed by inducing transient transitions of Σ_c that take Σ in a round trip from its initial state x^0 back to x^0, passing through critical indeterminate transitions. Such testing can involve only states from which there is a guaranteed return to the initial state $x^0 = x^j$, namely, only states of the set

$$\rho^0 := \{x^i \in X \mid K_{ij}(\Sigma) = 1\}. \qquad (14)$$

To test a pair (x, u), we must be able to reach x from the initial state $x^0 = x^j$, at least for some outcomes of indeterminate transitions, so x must be in the set

$$S^0 := \{x^i \in X \mid K_{ji}(\Sigma) \neq 0\}. \qquad (15)$$

During testing, after reaching a state x, it must be possible to return to the initial state x^0 from any outcome of a tested pair (x, u), so every tested pair (x, u) must be a member of the set

$$T(\Sigma, x^0) := \{(x, u) \in S^0 \times A \mid s(x, u) \subseteq \rho^0\}, \qquad (16)$$

where s is the stable recursion function of Σ. This leads to:

Proposition 17. Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with stable recursion function s. Then, an indeterminate pair of Σ can be tested in the initial state only if it belongs to the set $T(\Sigma, x^0)$ of (16). □

Note that not all members of $T(\Sigma, x^0)$ are testable in every sample of Σ. The pairs whose testing can be guaranteed a-priori are characterized by the following.

Proposition 18. Let (x, u) be an indeterminate pair of an asynchronous machine with skeleton matrix $K(\Sigma)$ and initial state $x^0 = x^j$. The following are equivalent.
(i) (x, u) can always be tested, irrespective of the outcomes of any indeterminate transitions.
(ii) (x, u) is a member of the set
$\tau(\Sigma, x^0) := \{(x^i, u) \in T(\Sigma, x^0) \mid K_{ji}(\Sigma) = 1\}$. □

We call $\tau(\Sigma, x^0)$ the set of *certainly testable pairs.*

Let A^i be as in (3). Then, the set of all pseudo characters associated with the set $T(\Sigma, x^0)$ of (16) is

$$A'(\Sigma, x^0) = \bigcup_{\{i \mid (z^i, u^i) \in T(\Sigma, x^0)\}} A^i. \qquad (17)$$

By Proposition 17, pseudo characters outside this set cannot be tested in the initial state. The set of pseudo characters that can be tested with certainty (namely, in all samples of Σ) is

$$A'_c(\Sigma, x^0) = \bigcup_{\{i | (z^i, u^i) \in \tau(\Sigma, x^0)\}} A^i. \tag{18}$$

The difference set $A'(\Sigma, x^0) \setminus A'_c(\Sigma, x^0)$ consists of pseudo characters that can be tested only in some samples of Σ.

Example 19. For the machine Σ of Example 2 with the initial state $x^0 = x^1$, a calculation shows that

$$A'(\Sigma, x^0) = A'_c(\Sigma, x^0) = A' = \{a^1, a^2, c^1, c^2\}. \ \square$$

Testing in the initial state is performed by input strings that take Σ from its initial state back to its initial state through indeterminate transitions. Considering that the state set, the input alphabet, and the pseudo alphabet are all finite sets, all such strings are of bounded length, say bounded by $\mu > 0$. Let $x^0 = x^\alpha$ be the initial state of Σ. Then, any family T of complete strings that take Σ through round trips from the initial state back to the initial state satisfies $T \subseteq (R^a(\Sigma, 1))^{(\mu)}_{\alpha\alpha}$. We have (see Yang, Xing, and Hammer [2011a] for a stronger statement):

Lemma 20. Let Σ be an indeterminate asynchronous machine with the initial state $x^0 = x^\alpha$ and the one-step matrix of stable transitions $R^a(\Sigma, 1)$. Then, there is an integer $\mu > 0$ such that the entry $(R^a(\Sigma, 1))^{(\mu)}_{\alpha\alpha}$ includes a family T of complete strings satisfying $A'_c(\Sigma, x^0) \subseteq \Pi' T$. $\square$

Example 21. For the machine Σ of Example 2 with $x^0 = x^1$, calculation yields:

Table 2. The family T of complete sets.

entry	The family T of complete sets (not all terms listed)	$\Pi' T$
$(R^a(\Sigma, 1))_{11}$	$\{\cdots\}$	$\varnothing$
$(R^a(\Sigma, 1))^{(2)}_{11}$	$\{\cdots\}$	$\varnothing$
$(R^a(\Sigma, 1))^{(3)}_{11}$	$\{a\gamma_1, a\gamma_2, a\gamma_3\}$	$\{c^1, c^2\}$
$(R^a(\Sigma, 1)^{(4)}_{11}$	$\left\{ \begin{array}{l} (R^a(\Sigma, 1))^{(3)}_{11}, a\gamma_1 b, a\gamma_2 b, \\ a\gamma_3 b, a\gamma_3 b, a\gamma_1 c, a\gamma_2 c, \\ a\gamma_3 c, ba\gamma_1, \cdots, ca\gamma_1, \cdots \end{array} \right\}$	$\{c^1, c^2\}$
$(R^a(\Sigma, 1)^{(5)}_{11}$	$\left\{ \begin{array}{l} (R^a(\Sigma, 1)^{(4)}_{11}, bca\gamma_1, bca\gamma_2, \\ bca\gamma_3, cba\gamma_1, \cdots, ad\gamma_4, \cdots \end{array} \right\}$	$\left\{ \begin{array}{c} a^1, a^2, \\ c^1, c^2 \end{array} \right\}$

where $\gamma_1 := c^1 + c^2 b$, $\gamma_2 := (c^1 + c^2)b$, $\gamma_3 := c^1 c + c^2 b$, and $\gamma_4 := a^1 \gamma_1 + a^2 b$ are complete sets. As $(R^a(\Sigma, 1)^{(5)}_{11}$ includes all pseudo characters, $\mu = 5$ in Lemma 20. $\square$

5.2 The Testing Algorithm

The next algorithm guides the controller as it tests the machine Σ in the initial state. The testing is automatic and returns Σ to its original initial state after testing. The testing constitutes a transient of Σ_c, and hence does not affect user experience. Recall that the true stable recursion function s_{true} indicates testing outcomes. An examination of the algorithm confirms the following (Yang, Xing, and Hammer [2011a]).

Theorem 22. Let Σ and Σ' be input/state asynchronous machines with identical state sets, where Σ' is determinate. Let $K(\Sigma)$ be the outcome of Algorithm 5.2. Then,

(i) Algorithm 5.2 can be implemented by a state feedback controller in fundamental mode operation.
(ii) Statements (a) and (b) are equivalent:

Testing in the Initial State: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with adjusted machine $\Sigma_a = (\tilde{A}, X, x^0, s_a)$, state set $X = \{x^1, ..., x^n\}$, initial state $x^0 = x^\alpha$, true stable recursion function s_{true}, one-step matrix of stable transitions $R^a(\Sigma, 1)$, and skeleton matrix $K(\Sigma)$. Let $\Sigma' = (A, X, x^0, s')$ be a determinate model with skeleton matrix $K(\Sigma')$, and let $S(\Sigma', \Sigma)$ be the reduced comparison matrix.

```
Step 0. Set $\beta := 0$ and $K(\Sigma, 0) := K(\Sigma)$.
Step 1. Let $A'(\Sigma, x^0)$ be given by (17). If no pseudo characters
    of $A'(\Sigma, x^0)$ appear in the matrix $S(\Sigma', \Sigma)$, then testing of
    indeterminate transitions in the initial state is not meaningful;
    go to Step 9.
Step 2. Let $T$ and $\mu$ be as in Lemma 20, and let $t_1, t_2, ..., t_q$ be
    the complete sets of strings included in $T$. Define the subsets
    of pseudo characters $\theta_i := \Pi' t_i, i = 1, 2, ..., q$. Replace $\beta$ by
    $\beta + 1$, set $j := 1$, and set $K(\Sigma, \beta) := K(\Sigma, \beta - 1)$.
Step 3. If $\theta_j$ has no characters in common with the matrix
    $S(\Sigma', \Sigma)$, then go to Step 7.
Step 4. Apply the complete set $t_j$ to the machine $\Sigma$ in funda-
    mental mode operation, as follows (after each character, wait
    until $\Sigma$ has reached its next stable state and record it):
    Based on the stable state reached by $\Sigma$, select the next input
    character $v$; if $v$ is a pseudo character, apply to $\Sigma$ the original
    character $u(v) \in A$ associated with $v$.
    Continue until $\Sigma$ returns to its initial state $x^0$.
Step 5. Partition the set $\theta_j$ into two disjoint subsets $\theta_j^+$ and $\theta_j^-$,
    where $\theta_j^+$ consists of all pseudo input characters $v \in \theta_j$ for
    which $s_{true}(z(v), u(v)) = s_a(z(v), v)$, and $\theta_j^- := \theta_j \setminus \theta_j^+$ (the
    set difference).
Step 6. Perform sub-steps (a) to (d) below for every pair of
    integers $p, \ell \in \{1, ..., n\}$:
    (a) If there is a member $\theta$ of $S_{p\ell}(\Sigma', \Sigma)$ satisfying $\theta \subseteq \theta_j^+$,
        then assign $K_{p\ell}(\Sigma, \beta) := 1$, and replace $S_{p\ell}(\Sigma', \Sigma)$ by the
        empty set.
    (b) Replace every member $\theta$ of $S_{p\ell}(\Sigma', \Sigma)$ by the difference
        set $\theta \setminus \theta_j^+$.
    (c) Remove from $S_{p\ell}(\Sigma', \Sigma)$ every member that is not disjoint
        with $\theta_j^-$. If this turns $S_{p\ell}(\Sigma', \Sigma)$ into the empty set, assign
        $K_{p\ell}(\Sigma, \beta) := 0$; then, if $K_{p\ell}(\Sigma') = 1$, set $K(\Sigma) := K(\Sigma, \beta)$
        and go to Step 9 (model matching is impossible).
    (d) If $K(\Sigma, \beta) \geq K(\Sigma')$, then set $K(\Sigma) := K(\Sigma, \beta)$ and go to
        Step 9 (model matching is possible).
Step 7. If $j < q$, replace $j$ by $j + 1$ and return to Step 3.
Step 8. If $K(\Sigma, \beta) \neq K(\Sigma, \beta - 1)$, then recalculate the one-
    step matrix of stable transitions $R^a(\Sigma, 1)$ and the comparison
    matrix $S(\Sigma', \Sigma)$. Set $K(\Sigma, \beta) := K(\Sigma)$ and Return to Step 1.
Step 9. Output the matrix $K(\Sigma)$ and terminate.
```

(a) Testing in the initial state determines that there is a controller C achieving $\Sigma_c = \Sigma'$ in fundamental mode operation.
(b) $K(\Sigma) \geq K(\Sigma')$.

(iii) If there is a pair of integers i, j for which $K_{ij}(\Sigma) = 0$ while $K_{ij}(\Sigma') = 1$, then there is no controller satisfying $\Sigma_c = \Sigma'$ in fundamental mode operation.
(iv) If there is a pair of integers i, j for which $K_{ij}(\Sigma) = \Delta$ while $K_{ij}(\Sigma') = 1$, then testing in the initial state cannot determine whether there exists of a controller C for which $\Sigma_c = \Sigma'$ in fundamental mode operation.

Example 23. Referring to Example 12, design a controller C for Σ to match Σ'. Assume that the true stable recursion function

s_{true} of Σ, which gives the results of the testing described below, is as in Table 3.

Table 3. The true stable recursion function.

state	a	b	c	d
x^1	x^2	x^1	$-$	$-$
x^2	x^2	$-$	x^4	x^3
x^3	x^2	$-$	$-$	x^3
x^4	$-$	x^1	x^4	x^4

Using Algorithm 5.2, we determine whether model matching is possible. Considering the matrix $S(\Sigma',\Sigma)$ of Example 15, a slight reflection shows that it is enough to test the pseudo characters $\{\{a^2\},\{c^2\}\}$. According to Example 21,

$$T = \{a(c^1+c^2b),\cdots,ad[a^1(c^1+c^2b)+a^2b],\cdots\} \\ =: \{t_1,\cdots,t_2,\cdots\};$$

by (17), we have $A'(\Sigma,x^0) = A'_c(\Sigma,x^0) = \{a^1,a^2,c^1,c^2\}$.

Step 0. Set $\beta := 0$ and $K(\Sigma,0) := K(\Sigma)$, where $K(\Sigma)$ is given in Example 2.

Step 1: As $A'(\Sigma,x^0)$ and $S(\Sigma',\Sigma)$ have pseudo characters in common, continue to Step 2.

Step 2: $\theta_1 := \Pi' t_1 = \{c^1,c^2\}$, and $\theta_2 := \Pi' t_2 = \{a^1,a^2,c^1,c^2\}$; set $j := 1$.

Step 3: According to Example 15, $\{\{a^2\},\{c^2\}\} \subseteq S_{14}(\Sigma',\Sigma)$. As θ_1 has the pseudo character c^2 in common with $S_{14}(\Sigma',\Sigma)$, proceed to the next step.

Step 4: In this Example, $u(c^1) = c$ and $u(c^2) = c$, namely, c is the real character corresponding to the pseudo characters c^1 and c^2. Using the complete set $t_1 = a(c^1+c^2b)$, operate Σ as follows: at the initial state $x^0 = x^1$, apply the input character a to reach the stable state x^2. Upon reaching x^2, apply to Σ the input character $c = u(c^1) = u(c^2)$, which activates an indeterminate transition of Σ. According to Table 3, the next stable state of Σ turns out to be x^4.

Step 5: Considering the adjusted stable recursion function s_a of Table 1, the true transition corresponds to the pseudo character c^2. Thus, $\theta_1^+ = \{c^2\}, \theta_1^- = \{c^1\}$.

Step 6: *(a)* Since $\theta_1^+ = \{c^2\}$, and $\{c^2\}$ is a member of $S_{14}(\Sigma',\Sigma)$ and of $S_{24}(\Sigma',\Sigma)$, set $K_{14}(\Sigma) = K_{24}(\Sigma) = 1$ and $S_{14}(\Sigma',\Sigma) = S_{24}(\Sigma',\Sigma) = \varnothing$. Then, (d) is valid, and model matching is possible. The algorithm terminates.

Remark 24. The same principles can be used to test indeterminate transitions in any stable state of Σ. These tests would encompass all testing possible in fundamental mode operation.

6. CONCLUSION

This note outlined an algebraic framework that yields adaptive controllers for asynchronous sequential machines with incompletely specified transitions. The controllers test the controlled machine to learn as much as possible about its unknown transitions without interfering with user experience. The information gained during testing improves closed loop performance through a process of controller adaptation.

REFERENCES

G. Barrett and S. Lafortune [1998], "Bisimulation, the supervisory control problem, and strong model matching for finite state machines, *Discrete Event Dyn. Systems: Theory & App.*, vol. 8, no. 4, pp. 377–429.

M. D. Dibenedetto, A. Saldanha and A. Sangiovanni-Vincentelli [1994], "Model matching for finite state machines," *Proc. IEEE Conf. on Decision and Control*, vol. 3, 1994, pp. 3117–3124.

X. J. Geng and J. Hammer [2005], "Input/output control of asynchronous sequential machines," *IEEE Tr. Aut. Control*, vol. 50, no. 12, pp. 1956–1970, 2005.

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

J. Hammer [1995], "On the modeling and control of biological signal chains," *Proc. IEEE Conf. on Decision and Control*, pp. 3747–3752, December 1995.

J. Hammer [1996a], "On the corrective control of sequential machines," *Int. J. Control*, vol. 65, no. 2, pp. 249–276.

J. HAMMER [1996b], "On the control of incompletely described sequential machines," *International Journal of Control*, vol. 63, no. 6, pp. 1005–1028.

R. Kumar, S. Nelvagal, and S. I. Marcus [1997], "A discrete event systems approach for protocol conversion," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 7, no. 3, pp. 295–315.

T. E. Murphy, X. J. Geng, and J. Hammer [2002], "Controlling races in asynchronous sequential machines," *Proc. IFAC World Cong.*, Barcelona, 2002.

T. E. Murphy, X. Geng, and J. Hammer [2003], "On the control of asynchronous machines with races," *IEEE Trans. Aut. Control*, vol. 48, no. 6, pp. 1073–1081, 2003.

P. J. G. Ramadge and W. M. Wonham [1987], "Supervisory control of a class of discrete event processes," *SIAM J. Control and Opt.*, vol. 25, no. 1, pp. 206–230.

M.-D. Shieh, C.-L. Wey, and P. D. Fisher [1993], "Fault effects in asynchronous sequential logic circuits," *IEE Proc.-E*, vol. 140, no. 6, pp. 327–332, 1993.

J. G. Thistle and W. M. Wonham [1994], "Control of infinite behavior of finite automata," *SIAM Journal on Control and Optimization*, vol. 32, no. 4, pp. 1075–1097.

N. Venkatraman and J. Hammer [2006b], "Controllers for asynchronous machines with infinite cycles," *Proc. 17th Int. Symp. on Mathematical Theory of Networks and Systems*, pp. 1002–1007, Kyoto, Japan, 2006.

N. Venkatraman and J. Hammer [2006c], "On the control of asynchronous sequential machines with infinite cycles," *Int. J. Control.*, vol. 79, no. 7, pp. 764–785, 2006.

J.-M. Yang and J. Hammer [2008a], "Counteracting the effects of adversarial inputs on asynchronous sequential machines," *Proceedings of the IFAC World Congress*, pp. 1432–1437, Seoul, Korea, July 2008.

J.-M. Yang and J. Hammer [2008b], "State feedback control of asynchronous sequential machines with adversarial inputs," *Int. J. Control.*, vol. 81, no. 12, pp. 1910–1929, 2008.

J.-M. Yang, T. Xing, and J. Hammer [2011] "On the control of indeterminate asynchronous sequential machines: an algebraic framework", *Proc. 2011 IEEE Int. Conf. on Intelligent Computing and Intelligent Systems (ICIS 2011)*, Guangzhou, China, November 2011.

J.-M. Yang, T. Xing, and J. Hammer [2011] "Adaptive control of asynchronous sequential machines with state feedback", *European J. Control (to appear)*.

N. Yevtushenko, T. Villa, R. Brayton, A. Petrenko, and A. Sangiovanni-Vincentelli [2008], "Compositionally progressive solutions of synchronous FSM equations," *Discrete Event Dyn. Sys.: Th. & Appl.*, vol. 18, no. 1, pp. 51–89.