Input/Output Control of Asynchronous Sequential Machines

Xiaojun Geng, *Member, IEEE,* and Jacob Hammer, *Fellow, IEEE*

***Abstract*—The problem of controlling a finite-state asynchronous sequential machine is examined. Main consideration is given to input/output control, where access to the state of the machine is not available. The objective is to use output feedback to control the machine so as to match a prescribed model. It is shown that necessary and sufficient conditions for the existence of appropriate controllers can be stated in terms of a simple comparison of two numerical matrices. Whenever controllers exist, algorithms for their design are outlined.**

***Index Terms*—Asynchronous sequential machines, model matching, observers, output feedback, separation principle.**

I. Introduction

Asynchronous sequential machines are important building blocks of high-speed digital computer and control systems. The present paper develops a methodology of controlling such machines and changing their behavior through the use of traditional feedback control techniques. Given an asynchronous finite-state machine Σ with undesirable behavior, we build around it an output feedback loop, with the objective of achieving desirable behavior of the closed-loop system. The control configuration is shown in Fig. 1.

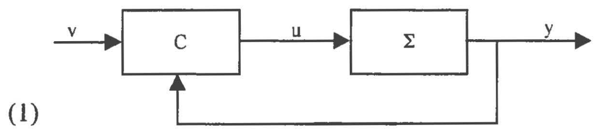


Fig. 1. Closed-loop system.

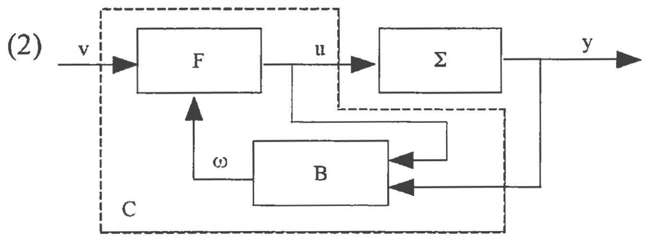


Fig. 2. Separation principle.

Here, Σ is the asynchronous machine being controlled and C is an asynchronous machine that serves as an output feedback controller. The objective is to design the controller C so that the closed loop system Σ_c exhibits desirable characteristics. We concentrate on the model matching problem, namely, the problem of finding a controller C for which the closed loop system Σ_c mimics a prescribed model Σ'. The techniques developed in the paper can also be used to address other problems related to the control of asynchronous machines.

The main results are presented in Section IV, which includes necessary and sufficient conditions for the existence of a controller C solving the model matching problem. To state the conditions, we associate with every asynchronous machine a numerical matrix called "skeleton matrix". The existence of C is then determined by comparing the "skeleton matrix" of the machine Σ to the "skeleton matrix" of the model Σ'. An algorithm for the construction of C is also provided.

In Section III, we prove the validity of a separation principle, whereby every solvable model matching problem for asynchronous machines can be solved by a controller C that decomposes into a combination an *observer* B and a state-feedback *control unit* F, as depicted in Fig. 2.

The observer B estimates the state of Σ from input/output data of Σ and feeds the estimated state to the control unit F. The latter generates a sequence that drives Σ along a desired path. To exhibit such a decomposition, we write $C = (F, B)$.

The framework developed here continues the process initiated in [1] to develop tools for the control of asynchronous machines. In [1], the model matching problem is considered for asynchronous machines whose state is available as output. When the state of the controlled machine is not available as output, the problem becomes more complex and requires the development of new concepts and new analytical tools.

Asynchronous machines are finite state sequential machines that operate without a clock; they are often called *clockless logic circuits*. Lacking a clock, an asynchronous machine is driven by changes of its input variables. A change of an input variable causes the machine go through a succession of state transitions. This succession of state transitions may or may not end. When it ends, the machine reaches a state at which it lingers until the next change of an input variable. This state and the current input value form a *stable combination* of the machine. When the succession of state transitions does not end, then the machine has an *infinite cycle*. Machines with infinite cycles are not discussed in this paper.

The intermediate states through which an asynchronous machine passes on its way to a stable combination are called *unstable states*. State transitions occur at the speed limit of the machine's components, as they are not slowed by waiting for the next clock tick. Consequently, asynchronous machines are used in some of the fastest computing equipment.

Manuscript received September 30, 2003; revised October 28, 2004. Recommended by Associate Editor L. Holloway.

X. Geng is with the Department of Electrical and Computer Engineering, California State University, Northridge, CA 91330 USA.

J. Hammer is with the Department of Electrical and Computer Engineering, the University of Florida, Gainesville, FL 32611 USA (e-mail: hammer@mst.ufl.edu).

Digital Object Identifier 10.1109/TAC.2005.860293

The lack of a synchronizing clock requires that special precautions be exercised when operating an asynchronous machine. Care has to be taken while the machine is in transition through a succession of unstable states. During such a transition, the exact state of the machine at any moment of time is unpredictable, since no clock governs the state changes, and the transition rate depends on unpredictable hardware conditions. Consequently, if an input change occurs during a transition process, the response of the machine can become unpredictable, since the state of the machine at the time of the input change is unpredictable. To avoid such uncertainty, it is common practice to keep the input of an asynchronous machine constant while the machine is not in a stable state. This leads to the following notion [2].

(3) Definition: A machine (or a composite machine) *operates in fundamental mode* if only a single variable of the machine can change its value at a timte. ♦

For a single machine, fundamental mode operation requires the rate of input changes not to exceed the response rate of the machine. For the composite system shown in Fig. 1, fundamental mode operation requires the input of Σ to remain constant while Σ undergoes a transition; and it requires the input of C to remain constant while C undergoes a transition. Clearly, the input of Σ remains constant while the output of C is constant, and the input of C remains constant while the output of Σ is constant (and the external input v is constant). This simple argument leads to the following [1].

(4) Proposition: Let Σ and C be asynchronous machines interconnected in the configuration of Fig. 1. Then, the configuration operates in fundamental mode if the following hold:

i) the output of C is constant while Σ is not in a stable combination; and
ii) the output of Σ and the input variable v are constant while C is not in a stable combination. ♦

All control configurations used in the present paper are designed to operate in fundamental mode.

The present discussion continues the investigation into the control of asynchronous sequential machines initiated in [1] and [3]. There, the model matching problem is considered for asynchronous machines whose state is available as output. When the state of the controlled machine is not available for output, as is the case here, the problem becomes more complex and requires the development of new tools.

The control of sequential machines has received considerable attention in the literature of the last two decades. Prominent are studies related to the supervisory control of discrete-event systems [4], [5], the references cited in these papers, and many others. In supervisory control, the objective is to develop a controller (or a supervisor) that elicits from the controlled machine a specified formal language. Other authors have studied sequential machines via the traditional control theoretic approach, concentrating on the development of controllers that resolve issues related to error correction, the reduction of disturbance effects, and model matching [6]–[12]. This literature on the control of sequential machines does not address certain issues that are critical to the operation of asynchronous machines, like the distinction between stable and unstable states and fundamental mode operation (see Section II for details). Addressing these issues requires new conceptual and analytical tools that form the subject of the present paper, of [1], and of [3]. The approach adopted here draws on a combination of tools taken from the traditional theory of digital design [2] and from traditional control theory.

	a	b	c	Y
x^1	x^4	x^3	x^1	0
x^2	x^2	x^2	x^3	1
x^3	-	x^2	x^3	0
x^4	x^2	x^3	x^4	1

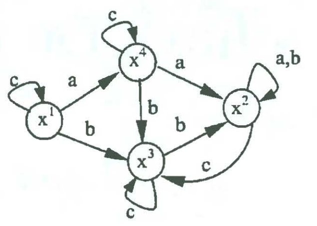


Fig. 3. Example.

The paper is organized as follows. Terminology and background are introduced in Section II. The decomposition of the controller into an observer and a state feedback is discussed in Section III. The solution of the model matching problem is presented in Section IV, where necessary and sufficient conditions for the existence of controllers are derived, and where design algorithms for controllers are developed. This paper concludes in Section V with a comprehensive example.

II. Terminology and Background

A. Asynchronous Machines and Stable Equivalence

An *asynchronous sequential machine* Σ is represented by a quintuple (A, Y, X, x_0, f, h), where A, Y, and X are nonempty finite sets—A is the input alphabet, Y is the output alphabet, and X is the state set; x_0 is the initial state of the machine, and $f : A \times X \to X$ and $h : X \to Y$ are partial functions, i.e., functions defined only over part of their domain. The machine Σ is represented by a recursion of the form

$$(5) \quad \begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= h(x_k), \qquad k = 0, 1, 2, \ldots \end{aligned}$$

Here, $u_0, u_1, u_2, \ldots$ is the *input sequence* of the machine; $y_0, y_1, y_2, \ldots$ is the *output sequence*; and $x_0, x_1, x_2, \ldots$ is the *state sequence*. The integer k represents the "step counter" of Σ; it advances by one upon a change of the machine's input or state. The function f is the *recursion function* (or state transition function) of Σ, and h is the *output function*. Note that the output function h of (5) does not depend on the input variable u, namely, that Σ is a Moore machine [2]. Considering that every asynchronous machine can be represented as a Moore machine, we observe that the use of the representation (5) is not restrictive. In the special case when the output function h is the identity function, Σ is called an *input/state machine*. An asynchronous machine can be represented by a graph or by a table of transitions, as indicated in the following.

(6) Example: An asynchronous machine with the input alphabet $A = \{a, b, c\}$, the output alphabet $Y = \{0, 1\}$, and the state set $X = \{x^1, x^2, x^3, x^4\}$ can be depicted by a graph or by a table of transitions. The transition function f and the output function h are depicted in the table. The symbol "-" in Fig. 3 indicates that the associated state-input pair is not to be used.

	a	b	c	Y
x^1	x^2	x^2	x^1	0
x^2	x^2	x^2	x^3	1
x^3	-	x^2	x^3	0
x^4	x^2	x^2	x^4	1

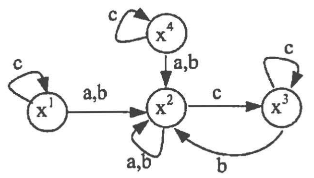


Fig. 4. Stable state machine.

(7) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine. A pair $(x, u) \in X \times A$ is a *valid pair* of Σ if the function f is defined at (x, u). The pair $(x, u) \in X \times A$ is a *stable combination* if $x = f(x, u)$. ♦

In Example 6, the pairs $(x^1, c), (x^2, a), (x^2, b)$, and (x^3, c) are stable combinations, while (x^4, a) is not. When (x, u) is not a stable combination, a machine Σ engages in a chain of transitions $x_1 = f(x, u), x_2 = f(x_1, u), \ldots$, which may or may not terminate. If this chain of transitions does not terminate, then Σ contains an *infinite cycle*. We assume throughout this paper that the machines under consideration have no infinite cycles. For such machines, the following concept is important.

(8) Definition: Let (x, u) be a valid pair of the machine $\Sigma = (A, Y, X, x_0, f, h)$, inducing the chain of transitions $x_1 = f(x, u), x_2 = f(x_1, u), \ldots$ Assume that there is an integer $i \geq 1$ for which (x_i, u) is a stable combination. Then, x_i is the *next stable state* of x with the input character u. ♦

Referring to Example 6, we see that x^2 is the next stable state of the state x^1 with the input a. The corresponding chain of transitions is $x^4 = f(x^1, a), x^2 = f(x^4, a), x^2 = f(x^2, a)$.

Ideally, when there are no infinite cycles, it takes zero time for an asynchronous machine to reach its next stable combination, irrespective of the number of intermediate transitions involved. Thus, from a user's point of view, only output values that correspond to stable combinations are noticeable, since the machine can linger only at stable combinations. The following recursion function therefore describes the "noticeable" behavior of the machine.

(9) Definition: For an asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$, let x' be the next stable state of a valid pair (x, u). The *stable recursion function* $s : X \times A \to X$ of Σ is given by $s(x, u) := x'$ for all valid pairs $(x, u) \in X \times A$. The *stable state machine* induced by Σ is represented by the quintuple (A, Y, X, x_0, s, h), and is denoted by $\Sigma_{|s}$. ♦

The stable state machine $\Sigma_{|s}$ describes the transitions of Σ between states at which the machine can linger, and thus it describes the behavior of Σ as seen by a user.

(10) Example: The machine of Example 6 induces the stable state machine shown in Fig. 4.

Recall that, in fundamental mode operation, only one variable of the machine can change at a time. Consequently, an input variable can change only after all state transitions have ceased, i.e., only after the machine has reached a stable combination. Consider an input string $u = u_0 u_1 \ldots u_{m-1}$ applied to a machine Σ at the initial state x_0. In fundamental mode operation, the first input value u_0 remains fixed until Σ reaches the next stable state $x_1 := s(x_0, u_0)$. Then, the input value switches to u_1 and stays constant until the next stable state $x_2 := (s(s(x_0, u_0), u_1)$ is reached. This process continues until the last stable state $x_m := s(\ldots s(s(s(x_0, u_0), u_1), u_2) \ldots, u_{m-1})$ is reached. We use the shorthand notation

$$s(x_0, u) := s(\ldots s(s(s(x_0, u_0), u_1), u_2) \ldots, u_{m-1}).$$

The notion of stable state machine leads to the following generalization of the classical notion of equivalence.

(11) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$, and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be two machines having the same input and the same output sets, and let $\Sigma_{|s}$ and $\Sigma'_{|s}$ be the stable state machines induced by Σ and Σ', respectively. Two states $x \in X$ and $\zeta \in X'$ are *stably equivalent* ($x \equiv \zeta$) if the following is true: When $\Sigma_{|s}$ starts from the state x and $\Sigma'_{|s}$ starts from the state ζ, then: i) $\Sigma_{|s}$ and $\Sigma'_{|s}$ have the same permissible input strings; and ii) $\Sigma_{|s}$ and $\Sigma'_{|s}$ generate the same output string for every permissible input string. The two machines Σ and Σ' are *stably equivalent* if their initial states are stably equivalent, i.e., if $x_0 \equiv \zeta_0$. ♦

Stably equivalent machines appear identical to a user.

B. Minimal Machines and Quotient Machines

The notion of equivalent states can also be applied to a single machine. A machine with equivalent states contains redundant states, and can be simplified by deleting one state of each pair of equivalent states. This leads to the following.

(12) Definition: A machine is *stably reduced* if its stable state machine has no states that are stably equivalent. ♦

The following notion was used in [1] and [3].

(13) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s, and let $\Sigma_{|s}$ be the stable state machine induced by Σ. A state x' of Σ is *stably reachable* from a state x of Σ (or x can stably reach x') if there is an input string u for which $x' = s(x, u)$. The machine Σ is *stably reachable* if every state of $\Sigma_{|s}$ is stably reachable from the initial state x_0. ♦

Another way of simplifying a machine without modifying its behavior is to remove inaccessible states. Removing all (stably) redundant and inaccessible states results in the next adaptation of a classical concept of automata theory [13].

(14) Definition: An asynchronous sequential machine is *stably minimal* if it is stably reduced and stably reachable. ♦

Another classical concept adapted to the present framework is the notion of a quotient machine. Recall that a partition $\Pi = \{\pi^1, \ldots \pi^m\}$ of a set X is simply a collection of disjoint subsets of X whose union is X. It is convenient to use the following notation. For a function $s : X \times A \to X$ and two subsets $S^1 \subset X$ and $S^2 \subset A$, denote by $[S^1, S^2] := S^1 \times S^2$ the cross product set, and by $s[S^1, S^2]$ the corresponding image, i.e., $s[S^1, S^2] := \{s(x, a) : x \in S^1, a \in S^2\}$.

(15) Definition: Let Σ be an asynchronous machine with the stable state machine $\Sigma_{|s} = (A, Y, X, x_0, s, h)$. Let X^r be the set of states that are stably reachable from the initial state x_0.

i) A partition $\Pi = \{\pi^1, \ldots, \pi^m\}$ of X^r is a *valid partition* when the following holds for every member $\pi \in \Pi$ and for every input character $u \in A$: If there is a state $x \in \pi$ for which (x, u) is a valid pair of Σ, then all elements of $\pi \times u$ are valid pairs of Σ. Then, $[\pi, u]$ is also called a *valid pair*.

ii) A valid partition Π is *transition consistent* if, for every valid pair $[\pi, u]$, there is a member $\pi' \in \Pi$ such that $s[\pi, u] \subset \pi'$.

iii) A valid partition Π is *output consistent* if the output function h is constant over each member of Π, i.e., if, for every $\pi \in \Pi$, one has $h(a) = h(b)$ for all states $a, b \in \pi$.

iv) The partition Π is a *stable equivalence partition* if it is valid, transition consistent, and output consistent. ♦

(16) Definition: Let Σ be an asynchronous machine with the stable state machine $\Sigma_{|s} = (A, Y, X, x_0, s, h)$, let X^r be the set of states stably reachable from the initial state x_0, and let $\Pi = \{\pi^1, \ldots, \pi^m\}$ be a stable equivalence partition of X^r. Then, the *stable quotient machine* Σ/Π of Σ with respect to Π is the machine $(A, Y, \Pi, \pi_0, s_\pi, h_\pi)$ whose recursion function s_π and output function h_π are defined as follows for every $\pi \in \Pi$:

i) $s_\pi(\pi, u) := \pi'$, where $\pi' \in \Pi$ is the member satisfying $s[\pi, u] \subset \pi'$;

ii) $h_\pi(\pi) := h(x)$, where x is an element of π.

The initial state π_0 of Σ/Π is the member of Π that contains the initial state x_0 of Σ. ♦

In intuitive terms, the stable quotient machine is a simpler machine whose stable state behavior is equivalent to that of Σ. The next definition and proposition follow along well-established lines of automata theory [13].

(17) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A', Y', X', \zeta_0, f', h')$ be two asynchronous machines with stable recursion functions s and s', respectively. Then, Σ and Σ' are *stably isomorphic* if there exist three set isomorphisms $\alpha : X \to X'$, $\beta : A \to A'$, and $\chi : Y \to Y'$ such that

i) $\alpha(s(x, u)) = s'(\alpha(x), \beta(u))$;

ii) $h(x) = \chi[h'(\alpha(x))]$;

iii) $\alpha(x_0) = \zeta_0$;

for all $x \in X$ and $u \in A$. ♦

(18) Proposition: Let Σ and Σ' be stably equivalent machines. If Σ' is stably minimal, then there is a stable equivalence partition Π for which the quotient machine Σ/Π is stably isomorphic to Σ'.

III. Detectability and Observers

A. Detectability of Asynchronous Machines

For the control configuration of Fig. 1 to operate in fundamental mode, the input value of Σ must remain constant while Σ undergoes state transitions. Considering that the input of Σ is the output of the controller C, this means that, after every change, the output of C must remain constant until Σ reaches a stable combination. Accordingly, C must determine whether or not Σ has reached its next stable combination. This determination must be based on the *input/output data* of Σ, i.e., on the input string and on the output string of Σ, since the controller has no access to the state of Σ.

To examine the situation more closely, let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the stable recursion function s. Assume that Σ is in a stable combination (x_1, v) when the input character changes to u. Let $x_1, x_2, \ldots, x_m$ be the string of states generated by this input change, where $x_m = s(x_1, u)$ is the next stable state of Σ, and, if $m > 1$, the other states satisfy $x_{i+1} = f(x_i, u), i = 1, \ldots, m-1$. The corresponding output string is then $h(x_1)h(x_2)\ldots h(x_m)$. When $m = 1$, the machine Σ remains in a stable combination, so there is no issue of detecting arrival at the next stable state. However, when $m > 1$, caution is required, since the output values $h(x_1), h(x_2), \ldots, h(x_m)$ generated during this transition may not be all distinct. In an asynchronous environment, it is impossible to distinguish between consecutive equal values of a string, since there is nothing to mark the start of a new step. In other words, constant segments of the output string appear as a single element. The presence of a constant segment in the output string may create a difficulty in recognizing arrival at the next stable state. For example, when $h(x_i) = h(x_{i+1})$ for an $i \in \{1, \ldots, m-1\}$, it is impossible to determine from input/output data when (or whether) the transition from x_i to x_{i+1} has occurred. (The input data does not help in this regard, since the input character is kept fixed during a chain of transitions.) The following concepts are crucial in this context.

(19) Definition: An asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ is *detectable* at a stable combination (x, u) if it is possible to determine from input/output data whether Σ has reached the next stable state x' of (x, u); if so, the transition from (x, u) to x' is called a *stable and detectable transition.*

(20) A stably reachable machine $\Sigma = (A, Y, X, x_0, f, h)$ is *detectable* if every state of Σ can be reached from the initial condition x_0 through a chain of stable and detectable transitions. ♦

(21) Definition: Let Y be an alphabet and let $y_1, \ldots, y_q \in Y$ be a list of characters such that $y_{i+1} \neq y_i$ for all $i = 1, \ldots, q-1$. Then, the *burst* of a string $y = y_1 y_1 \ldots y_1 y_2 y_2 \ldots y_2 \ldots y_q y_q \ldots y_q$ is $\beta(y) := y_1 y_2 \ldots y_q$. ♦

The burst is obtained by replacing each segment of repeating characters by a single occurrence of the same character. For example, the string $abbcccaa$ has the burst $\beta(abbcccaa) = abca$. The burst is the only discernible entity of an asynchronous output string. When referring to the output string generated by Σ from the valid pair (x_1, u) discussed earlier, we denote the resulting burst by the shorter notation

$$(22) \quad \beta(x_1, u) := \beta(h(x_1)h(x_2)\ldots h(x_{m-1})h(x_m)).$$

Let us reconsider now the situation described in the paragraph preceding Definition 19. In order to determine from input/output data whether Σ has reached the stable combination (x_m, u), it must be possible to determine whether the output string $h(x_1)h(x_2)\ldots h(x_m)$ has reached its end. This, however, is not always feasible. Consider, for example, the case where $m = 3$, and $h(x_1) = a, h(x_2) = b, h(x_3) = b$. Here, it is not possible to determine from the output whether the machine has reached the stable combination (x_3, u), since the output switches to b at the state x_2 and remains unchanged during the transition from x_2 to x_3. The difficulty originates from the fact that one observes only the burst of a string, not the string itself. Consequently, the end of the output string can be determined if and only if it is signified by a difference in the burst, i.e., if and only if $\beta(h(x_1)h(x_2)\ldots h(x_{m-1})) \neq \beta(h(x_1)h(x_2)\ldots h(x_{m-1})h(x_m))$. The latter is equivalent to $h(x_{m-1}) \neq h(x_m)$. This provides a practical test for detectability and justifies the next statement. We use the notation

$$\beta_{-1}(x_1, u) := \begin{cases} \beta(h(x_1)h(x_2)\ldots h(x_{m-1}) & \text{for } m > 1 \\ \oslash \quad \text{for } m = 1. \end{cases}$$

(23) Proposition: An asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$ is detectable at a valid pair (x, u) if and only if $\beta_{-1}(x, u) \neq \beta(x, u)$.

(24) Example: For the machine Σ of Example 6, the pair (x^1, a) induces the chain of state transitions x^1, x^4, x^2, where (x^2, a) is the next stable combination. The corresponding output string is $h(x^1)h(x^4)h(x^2) = 011$, for which we have $\beta(x^1, a) = \beta_{-1}(x^1, a) = 01$. Therefore, the machine Σ is not detectable at (x^1, a). A similar examination shows that Σ is detectable at (x^1, b). ♦

In general, of course, an asynchronous machine may or may not be detectable at any one of its valid pairs. However, for input/state machines, the situation is different.

(25) Proposition: An asynchronous input/state machine is detectable at all its valid pairs.

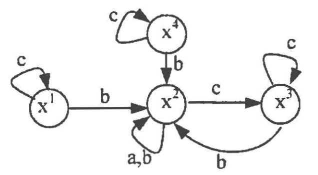
Proof: We use the notation of the previous paragraphs regarding the valid pair (x_1, u). When $m = 1$, one always has $\beta_{-1}(x_1, u) \neq \beta(x_1, u)$, since $\beta_{-1}(x_1, u) = \oslash$ and $\beta(x_1, u) = x_1$. When $m > 1$, the states $x_1, x_2, \ldots, x_m$ must all be distinct—a repeating state in this list would either create a stable combination before x_m is reached, or it would cause an infinite loop. But then, $\beta_{-1}(x_1, u) = x_1x_2\ldots x_{m-1}$, while $\beta(x_1, u) = x_1x_2\ldots x_m$, so $\beta_{-1}(x_1, u) \neq \beta(x_1, u)$. ♦

B. Skeleton Matrices

In [1] and [3], it was shown that the control capabilities of an input/state machine Σ are determined by its *skeleton matrix* $K(\Sigma)$. In brief, $K(\Sigma)$ is a matrix of zeros and ones, whose (i, j) entry $K_{ij}(\Sigma) = 1$ if the state x^j is stably reachable from the state x^i; otherwise $K_{ij}(\Sigma) = 0$. Presently, we deal with input/output asynchronous machines, and this requires a refinement of the notion of skeleton matrix. Specifically, detectability has to be taken into consideration.

(26) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$ be a machine with state set $X = \{x^1, \ldots, x^n\}$ and stable recursion function s. The *one-step fused skeleton matrix* $D(\Sigma)$ is an $n \times n$ matrix of zeros and ones whose (i, j) entry is shown in the equation at the bottom of the page. ♦

(27) Example: Consider the machine Σ of Example 6. The state x^1 can stably reach x^2 with the input characters a and b. However, by Example 24, the machine is not detectable at (x^1, a). As the machine is detectable at (x^1, b), the entry



$$D(\Sigma) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} ♦$$

Fig. 5. Fussed skeleton matrix.

$D_{12}(\Sigma)$ is 1. Fig. 5 illustrates all one-step detectable and stably reachable transitions of Σ.

Here is a systematic way of computing the one-step fused skeleton matrix.

(28) Algorithm: Let $\Sigma = (A, Y, X, x_0, f, h)$ be an asynchronous machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and input alphabet $A = \{u^1, \ldots, u^m\}$, where n is the number of states and m is the number of input characters. Let s be the stable transition function of Σ. The following steps derive the $n \times n$ one-step fused skeleton matrix $D(\Sigma)$.

Start: Set $i := 1, q := 1$, and set all entries of $D(\Sigma)$ to zero.

Step 1) If $i = n$ and $q = m + 1$, then the Algorithm terminates, and the current value of $D(\Sigma)$ is the one-step fused skeleton matrix of Σ; otherwise, continue.

Step 2) If $q > m$, then increase the value of i by 1, and set $q := 1$; otherwise, continue.

Step 3) If (x^i, u^q) is a valid pair of s, then let j be the integer for which $s(x^i, u^q) = x^j$, and continue to Step 4); otherwise, increase the value of q by 1 and go to Step 1).

Step 4) Using Proposition 23, determine whether Σ is detectable at the pair (x^i, u^q). If yes, set $D_{ij}(\Sigma) := 1$, where j is from Step 3; otherwise, leave $D(\Sigma)$ unchanged. Increase the value of q by 1, and go to Step 1). ♦

(29) Remark: Algorithm 28 has polynomial complexity. Indeed, letting n be the number of states and m the number of input characters, it follows that the total number of state-input pairs is mn. Finding the next stable state of each pair requires checking a maximum of n transitions. Verifying whether a stable transition is detectable requires checking a maximum of n output values. Thus, the total number of steps in the calculation of the matrix $D(\Sigma)$ cannot exceed n^3m. ♦

In the special case when Σ is an input/state machine, it follows by Proposition 25 that the one-step fused skeleton matrix is equal to the one-step skeleton matrix of [1] and [3]. However, in general, the two matrices are not always equal.

We review next some operations on skeleton matrices (see [1] for more details). Let A, B be two $n \times n$ matrices of zeros and ones. The *combination* AB is again an $n \times n$ matrix of zeros and ones; its (i, j) entry is $(AB)_{ij} := \max\{A_{ik}B_{kj} : k =$

$$D_{ij}(\Sigma) = \begin{cases} 1, & \text{if there is a character } u \in A \text{ such that } \Sigma \text{ is} \\ & \text{detectable at } (x^i, u) \quad \text{and} \quad x^j = s(x^i, u) \\ 0, & \text{otherwise } i, j = 1, \ldots, n \end{cases}$$

$1, \ldots, n\}$ for all $i, j = 1, \ldots, n$. With combination, we can consider the kth "power" $D^k(\Sigma)$ of the one-step fused skeleton matrix, $k = 1, 2, \ldots$ Let $D^k_{ij}(\Sigma)$ be the (i, j) entry of $D^k(\Sigma)$. Define the matrix $D^{(m)}(\Sigma)$ by setting its (i, j) entry to

$$D^{(m)}_{ij}(\Sigma) := \max_{k=1,\ldots,m} D^k_{ij}(\Sigma), \qquad m = 1, 2, \ldots$$

Then, $D^{(m)}(\Sigma)$ is also a matrix of zeros and ones, and $D^{(1)}(\Sigma) = D(\Sigma)$. The following terminology is convenient.

(30) Definition: Let x^i, x^j be states of the asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$, and let $t > 1$ be an integer. A *chain of* t *detectable stable transitions* from x^i to x^j is a string of detectable pairs $(x^i, u_1), (x_2, u_2), \ldots, (x_{t-1}, u_{t-1}), (x_t, u_t)$, where $u_1, u_2, \ldots, u_t \in A$ are input characters, and where $x_2 := s(x^i, u_1), x_3 := s(x_2, u_2), \ldots, x_t := s(x_{t-1}, u_{t-1})$, and $x^j = s(x_t, u_t)$. ♦

Note that, in the Definition, $(x_2, u_1), (x_3, u_2), \ldots, (x_t, u_{t-1}), (x^j, u_t)$ are all stable combinations.

Regarding the matrix $D^{(m)}(\Sigma)$, a slight reflection shows that $D^{(m)}_{ij} = 1$ if and only if the state x^j can be reached from the state x^i through a chain of m or fewer detectable stable transitions. The case $m = n - 1$, where n is the number of states of Σ, is of particular importance.

(31) Definition: Let $D(\Sigma)$ be the $n \times n$ one-step fused skeleton matrix of the asynchronous machine Σ. The *fused skeleton matrix* of Σ is $\Delta(\Sigma) := D^{(n-1)}(\Sigma)$. ♦

The special significance of the case $m = n - 1$ originates from the following fact, whose proof is analogous to the proof of [1, Lemma 3.9]: If x^j cannot be reached from x^i within $n-1$ detectable stable transitions, then x^j cannot be reached from x^i in any number of detectable stable transitions. This yields the next result.

(32) Proposition: Let x^i and x^j be two states of the asynchronous machine Σ, and let $\Delta(\Sigma)$ be the fused skeleton matrix of Σ. The following two statements are equivalent:

i) there is a chain of detectable stable transitions from the state x^i to the state x^j;

ii) $\Delta_{ij}(\Sigma) = 1$.

(33) Example: For the one-step fused skeleton matrix $D(\Sigma)$ of Example 27, a simple calculation shows that

$$\Delta(\Sigma) = D^3(\Sigma) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}.$$

In the special case of an asynchronous input/state machine, it follows by Proposition 25 that the fused skeleton matrix is equal to the skeleton matrix of [1] and [3].

Recall that fundamental mode operation implies that the controller C of Fig. 1 can guide the system Σ only along paths that are chains of detectable stable transitions. When this fact is combined with Proposition 32, it gives rise to the expectation that fused skeleton matrices would be critical in determining the control capabilities of asynchronous machines. The forthcoming sections bear out this expectation.

C. Observers

The notion of an observer is employed here in a way that is similar to its use in other branches of control theory. In the present context, an observer is an asynchronous input/state machine whose purpose is to calculate the present state of another asynchronous machine, using input/output data of that machine. In addition, the observer must also indicate when the observed machine has reached its next stable state.

For a machine $\Sigma = (A, Y, X, x_0, f, h)$, one can attempt to build an observer by using the input/state part of Σ, given by $\Sigma_f := (A, X, X, x_0, f, I)$; here I indicates the identity function. The machine Σ_f does reproduce, of course, the transitions of the input/state part of Σ, except that the transitions of Σ_f and Σ are not synchronized. For example, assume that, in response to an input string, the machine Σ passes through a state x; then, the machine Σ_f must also pass through x in response to the same input string. However, since the machines are asynchronous, Σ may reach the state x either before, or during, or after the time at which x is reached by Σ_f. In other words, due to lack of synchronization, Σ_f cannot reproduce the current state of Σ. In fact, this argument indicates that it is not possible to build an observer for transient states of Σ; as the machines (ideally) spend zero time in a transient state, there is no opportunity to synchronize them in such a state. Thus, the most one can hope for is to build an observer that reveals stable combinations of Σ and indicates whether Σ has reached a stable combination. The latter is possible only for detectable transitions. The next statement introduces an auxiliary function that helps build an observer for Σ.

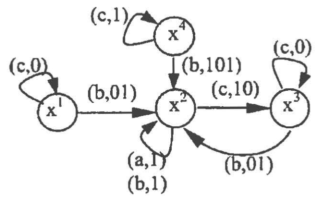
(34) Lemma: Let (x_1, u) be a valid pair of the asynchronous machine $\Sigma = (A, Y, X, x_0, f, h)$, and assume that the next stable state x_m of (x_1, u) is reached after $m - 1$ steps. When $m > 1$, define the chain of transitions $x_{i+1} = f(x_i, u), i = 1, \ldots, m-1$ and the burst $\beta_j := \beta(h(x_1)h(x_2)\ldots h(x_j)), j \in \{1, \ldots, m\}$. Then the following two statements are equivalent:

i) the machine Σ is detectable at the pair (x_1, u);

ii) there is a function $\kappa(x_1, u, \bullet) : Y^* \to \{0, 1\}$ such that $\kappa(x_1, u, \beta_j) = 1$ if and only if $j = m$. Here, Y^* is the set of all bursts of characters of Y.

Proof: Using the notation of (22), we have $\beta_m = \beta(x_1, u)$. Now, assume that (ii) is valid. Then, it must be true that $\kappa(x_1, u, \beta_{-1}(x_1, u)) \neq \kappa(x_1, u, \beta(x_1, u))$. The latter implies that $\beta_{-1}(x_1, u) \neq \beta(x_1, u)$, and part i) follows by Proposition 23. Conversely, assume that i) is valid. Using again Proposition 23, we have that $\beta_{-1}(x_1, u) \neq \beta(x_1, u)$. Then, the function $\kappa(x_1, u, \bullet) : Y* \to \{0, 1\}$ defined by $\kappa(x, u, \beta(x_1, u)) := 1$ and $\kappa(x_1, u, y) := 0$ for all $y \neq \beta(x_1, u)$ satisfies condition ii), completing the proof. ♦

Now, we can build an observer that reproduces all stable and detectable transitions of the machine Σ. The observer for Σ is an input/state machine $B = (A \times Y^*, X, Z, z_0, \sigma, I)$ with two inputs: the input character $u \in A$ of Σ and the output burst $\beta \in Y^*$ of Σ; the state set Z is identical to the state set X of Σ, and the initial condition is identical to that of Σ, i.e., $z_0 = x_0$. The recursion function $\sigma : Z \times A \times Y* \to Z$ of B is constructed as



♦

Fig. 6. Observer.

follows. First, using the stable recursion function s of Σ, define the function $\lambda : Z \times A \times \{0,1\} \to Z$ by

$$(35) \quad \lambda(z,u,a) := \begin{cases} s(z,u), & \text{if } a = 1 \\ z, & \text{if } a = 0. \end{cases}$$

Next, assume that Σ is in a stable combination (x, u_{i-1}) when the input character changes to u_i, where (x, u_i) is a detectable pair. The change of the input character may give rise to a chain of transitions of Σ. Let $k \geq i$ be a step during this chain of transitions, let β_k be the burst of Σ from step i to step k, and let u_k be the input character of Σ at step k. By fundamental mode operation, the input is kept constant during a chain of transitions, so we have $u_k = u_i$. Employing the function κ of Lemma 34 with the state $z = x$, define

$$(36) \quad \sigma(x, u_k, \beta_k) := \lambda(x, u_k, \kappa(x, u_k, \beta_k))$$

or, explicitly

$$(37) \quad \sigma(x, u_k, \beta_k) := \begin{cases} s(x, u_k), & \text{if } \beta_k = \beta(x, u_k) \\ x, & \text{otherwise.} \end{cases}$$

The observer B is then an input/state machine defined by

$$(38) \quad B: \quad \begin{cases} z_{k+1} = \sigma(z_k, u_k, \beta_k) \\ \omega_k = z_k \end{cases}$$

where z_k and ω_k are the state and the output of B at the step k, respectively; B is a stable state machine.

To describe the operation of the observer, assume that the observer switched to the state x immediately after Σ has reached the stable combination (x, u_{i-1}). Let $p \geq i$ be the step at which the chain of transitions from (x, u_i) to the next stable state $x' = s(x, u_i)$ terminates; then, $\beta_p = \beta(x, u_i)$. As the pair (x, u_i) is detectable, it follows by the definition of σ that the output of the observer B switches to the state x' at the step $p+1$. This leads to the following statement.

(39) Proposition: Let Σ be an asynchronous machine, and let B be an observer of Σ given by (38). Then, B displays as its output the most recent stable state that Σ has reached through a detectable transition. ♦

In view of our earlier discussion and Lemma 34, it is not possible to infer the state of Σ if it is not a stable state reached through a detectable transition.

(40) Example: An observer for the machine Σ of Example 6 is described by Fig. 6 (see also Example 27).

We can now summarize the implications of our discussion on the control configuration of Fig. 1. By fundamental mode operation, the output of the controller C must remain constant while the machine Σ is in transition. To fulfill this requirement, it must be possible for C to determine whether Σ has completed its transition process. As discussed, the end of a transition process can be detected only for detectable transitions. For a detectable transition, the output of the observer B switches to the next stable state of Σ immediately after Σ has reached that state; this signifies the end of the transition process and indicates the most recent stable state of Σ. In this way, the observer B helps create an environment in which the machine Σ can be controlled in fundamental mode operation.

These considerations lead us to the conclusion that only detectable stable transitions can be utilized when controlling a machine Σ. The fact that all such transitions are characterized by the fused skeleton matrix $\Delta(\Sigma)$ explains the significance of this matrix to the control of asynchronous machines.

IV. Controllers: Existence and Design

In this section, we derive necessary and sufficient conditions for the existence of a controller solving the model matching problem. When such a controller exists, we show that it can always be designed as a combination of an observer and a control unit, as depicted in Fig. 2. The structure of the observer was described earlier in (38), and an algorithm for the construction of the control unit is developed later in this section. First, a technical notion that is critical to the sequel.

A. Fused Skeleton Matrices and Reachability Indicators

(41) Definition: Let Σ be an asynchronous machine with the state set X, and let Λ^1 and Λ^2 be two nonempty subsets of X. The *reachability indicator* $r(\Sigma, \Lambda^1, \Lambda^2)$ is 1 if every element of Λ^1 can reach an element of Λ^2 through a chain of stable and detectable transitions; otherwise, $r(\Sigma, \Lambda^1, \Lambda^2) := 0$. ♦

The reachability indicator can be easily calculated from the fused skeleton matrix as follows. Let Σ be an asynchronous machine with state space X and fused skeleton matrix $\Delta(\Sigma)$. Let Λ^1 and Λ^2 be nonempty subsets of X, where Λ^1 has m elements and Λ^2 has p elements. Build the $m \times p$ matrix $\Delta_{\Lambda^1,\Lambda^2}(\Sigma)$ by deleting from $\Delta(\Sigma)$ all rows that correspond to states not in Λ^1 and all columns that correspond to states not in Λ^2. Next, create a column vector V by adding all columns of $\Delta_{\Lambda^1,\Lambda^2}(\Sigma)$. Then, a slight reflection shows that $r(\Sigma, \Lambda^1, \Lambda^2) = 1$ if and only if V has no zero entries. Here is an example.

(42) Example: Consider a machine Σ with the state set $X = \{x^1, x^2, x^3\}$ and the fused skeleton matrix

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

Let $\Lambda^1 = \{x^1, x^2\}$ and $\Lambda^2 = \{x^1, x^3\}$; then

$$\Delta_{\Lambda^1,\Lambda^2}(\Sigma) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad V = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

so that $r(\Sigma, \Lambda^1, \Lambda^2) = 1$. ♦

Since every state has a stable combination, $r(\Sigma, \Lambda^1, \Lambda^1) = 1$ for every nonempty subset Λ^1. Next, we extend the notion of a fused skeleton matrix to lists of subsets of the state set.

(43) Definition: Let Σ be a machine with the state set X, and let $\Lambda = \{\Lambda^1, \ldots, \Lambda^m\}$ be a list of $m \geq 1$ nonempty subsets of

X. The *fused skeleton matrix* $\Delta(\Sigma, \Lambda)$ of Λ is an $m \times m$ matrix whose (i, j) entry is $\Delta_{ij}(\Sigma, \Lambda) := r(\Sigma, \Lambda^i, \Lambda^j)$. ♦

Continuing with Example 42 and setting $\Lambda := \{\Lambda^1, \Lambda^2\}$, we obtain by a simple entry-by-entry calculation that

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

♦

In the special case where $\Lambda^i = \{x^i\}, i = 1, \ldots, n$, we get $\Delta(\Sigma, \Lambda) = \Delta(\Sigma)$. Our ensuing discussion revolves around lists of subsets, so we introduce now some terminology in this context.

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

B. Output Equivalence Lists

Consider the control of an asynchronous machine $\Sigma = (A, X, Y, x_0, f, h)$ in the configuration of Fig. 1. Here, the output string y is determined by the input string v through the equations

$$u = C(v, y)$$
$$y = \Sigma u.$$

As we can see, the controller C has two inputs: one is the external input v of the closed loop system and the other one is the output burst y of the controlled machine Σ. To simplify notation, assume that the input alphabet of the closed loop system is the same as the input alphabet A of Σ. As the bursts of Σ are elements of the set $Y*$ of output character strings, the input set of C is the cross product set $A \times Y^*$. We take the output set of C to be A, the input set of Σ, since the output of C is connected to the input of Σ. Letting T be the state set, ψ the recursion function, μ the output function, and t_0 the initial state of C, we can write $C = (A \times Y*, A, T, t_0, \psi, \mu)$.

Then, the closed loop system Σ_c of Fig. 1 has the input set A and the output set Y; being a combination of the machines C and Σ_c its state set is $T \times X$. Denoting by s_c the stable recursion function of Σ_c and by h_c the output function, we can write $\Sigma_c = (A, Y, T \times X, (t_0, x_0), s_c, h_c)$. As the output of Σ_c is the output of Σ, the output function $h_c : T \times X \times A \to Y$ satisfies $h_c((t, x), v) = h(x)$ for all states $(t, x) \in T \times X$ and for all input characters $v \in A$. The following is a common term.

(45) Defintion: The closed loop machine Σ_c is *well posed* if its output string is uniquely determined by the input string and the initial conditions. ♦

When the controller C consists of a combination (F, B) of an observer B and a control unit F, as depicted in Fig. 2, the composite system is represented by the following equations:

$$u = F(v, \omega)$$
$$\omega = B(u, y)$$
$$y = \Sigma u.$$

The control unit F has two inputs: the external input v of the configuration and the output ω of the observer, so its input set is $A \times X$. The output of F is the input of Σ, so the output set of F is A. Denoting by Ξ the state set of F, by ξ_0 its initial condition, by ϕ its recursion function, and by η its output function, we obtain $F = (A \times X, A, \Xi, \xi_0, \phi, \eta)$. The observer $B = (A \times Y^*, X, Z, z_0, \sigma, I)$ is given by (38).

In this case, the controller $C = (A \times Y^*, A, T, t_0, \psi, \mu)$ is a combination of F and of B. Consequently, the controller's state set T is the direct product of the state set Ξ of F and of the state set Z of B, namely, $T = \Xi \times Z$; its recursion function is $\psi = \phi \times \sigma$, and its initial condition is $t_0 = (\xi_0, x_0)$. As the output of C is the output of F, the controller's output function satisfies $\mu(\xi, z) = \eta(\xi)$ for all $(\xi, z) \in T$.

We turn now to a formal statement of the model matching problem. Let Σ be a machine that exhibits undesirable behavior. Assume that the desirable behavior is specified by an asynchronous machine Σ'. It is then necessary to design a controller C for which the behavior of the closed loop system Σ_c simulates the behavior of Σ'. The machine Σ' is then called the *model.* As discussed earlier (see also [2]), the practical performance of an asynchronous machine is determined by its stable-state behavior. Thus, from a practical standpoint, "the behavior of Σ_c simulates the behavior of Σ'" when the stable-state behavior of Σ_c is equivalent to the stable-state behavior of Σ'. This leads to the following.

(46) The Model Matching Problem: Given a machine Σ and a model Σ', find necessary and sufficient conditions for the existence of a controller C such that Σ_c is stably equivalent to Σ' and operates in fundamental mode. If such a controller exists, derive an algorithm for its design. ♦

Considering that only the stable state behavior of the model is relevant to the model matching problem, and that reduction to stably minimal form does not alter the stable-state behavior, it follows that the model Σ' can always be taken as a stably minimal machine. The model matching problem concentrates on matching the stable input/output behavior of the model.

The next definition introduces a notion which underlies our solution of the model matching problem for asynchronous machines. First, some notation. Given two sets S^1 and S^2 and a function $g : S^1 \to S^2$, denote by g^I the inverse set function of g; explicitly, for an element $s \in S^2$, the value $g^I(s)$ is the set of all elements $\alpha \in S^1$ satisfying $g(\alpha) = s$.

(47) Definition: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be two asynchronous machines with the same input set and the same output set, where the state set X' of Σ' consist of the q states $\zeta^1, \ldots, \zeta^q$. Define the subsets $E^i := h^I h'(\zeta^i) \subset X, i = 1, \ldots, q$. Then, $E(\Sigma, \Sigma') := \{E^1, \ldots, E^q\}$ is the *output equivalence list* of Σ with respect to Σ'. ♦

In other terms, an equivalence list is characterized by the following property: the value of the output function h of Σ at any state of the set E^i is equal to the value of the output function h' of Σ' at the state ζ^i. The members of an output equivalence list are not necessarily disjoint sets.

(48) Example: Let $X = \{x^1, x^2, x^3\}$ be the state set of the machine Σ, and let the output function h of Σ be given by $h(x^1) = 0, h(x^2) = 0, h(x^3) = 1$. Assume that the machine

Σ' has the state set $X' = \{\zeta^1, \zeta^2\}$ and the output function h' given by $h'(\zeta^1) = 0, h'(\zeta^2) = 1$. Then, the output equivalence list of Σ with respect to Σ' is $E(\Sigma, \Sigma') = \{E^1, E^2\} = \{\{x^1, x^2\}, \{x^3\}\}$. ♦

C. Existence of Controllers

We are now ready to address the model matching problem, namely, to construct a controller C for which Σ_c is stably equivalent to a specified model Σ'. The first step in this direction is the following statement, which originates from the fact that the stable equivalence of Σ_c and Σ' gives rise to an equivalence partition Π of the states of Σ_c, under which the quotient machine Σ_c/Π is equivalent to Σ' (Proposition 18). The statement provides a preliminary glimpse of the significance of the fused skeleton matrix, and we show later that its converse direction is also true. Regarding notation, let A and B be two $p \times q$ numerical matrices. The expression $A \geq B$ indicates that every entry of the matrix A is not less than the corresponding entry of the matrix B, i.e., $A_{ij} \geq B_{ij}$ for all $i = 1, \ldots, p$ and for all $j = 1, \ldots, q$.

(49) Lemma: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be asynchronous machines, where Σ' is stably minimal. Let $X' = \{\zeta^1, \ldots, \zeta^q\}$ be the state set of Σ', where the initial condition of Σ' is $\zeta_0 = \zeta^d$. Assume that there is a controller C for which Σ_c is stably equivalent to Σ' and operates in fundamental mode. Then, there is a nondeficient subordinate list Λ of the output equivalence list $E(\Sigma, \Sigma')$ for which $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.

Proof: Let C be a controller for which the composite machine $\Sigma_c = (A, Y, \Xi \times X, (\xi_0, x_0), s_c, h_c)$ is stably equivalent to the model $\Sigma' = (A, Y, X', \zeta_0, s', h')$, where Σ' has the state set $X' = \{\zeta^1, \ldots, \zeta^q\}$. Recall that the state set of Σ_c consists of all pairs (ξ^i, x^j), where ξ^i is a state of the controller C and x^j is a state of Σ. The initial state of Σ_c is (ξ_0, x_0), where ξ_0 is the initial state of C and x_0 is the initial state of Σ. Let G the set of all states of Σ_c that are stably reachable from the initial state (ξ_0, x_0). By Proposition 18, there is an equivalence partition $\Pi = \{\pi^1, \ldots, \pi^q\}$ of G for which the quotient machine Σ_c/Π is stably isomorphic to Σ'. This partition induces the stable equivalences $\pi^i \equiv \zeta^i, i = 1, \ldots, q$. Each member π^i of Π is not empty and consists of a subset of G; for reference, write

$$\pi^i = \left\{\left(\xi^{\sigma(i,1)}, x^{\delta(i,1)}\right), \left(\xi^{\sigma(i,2)}, x^{\delta(i,2)}\right), \ldots, \left(\xi^{\sigma(i,m(i))}, x^{\delta(i,m(i))}\right)\right\} \subset G$$

where $m(i)$ is the number of elements of π^i, and where $\sigma(i, j)$ and $\delta(i, j)$ are appropriate integers. Denote by π_0 the member of Π that is equivalent to the initial state ζ_0 of Σ'. Then, π_0 is the initial state of the quotient machine Σ_c/Π, so that $(\xi_0, x_0) \in \pi_0$; since $\zeta_0 = \zeta^d$, we have $\pi_0 = \pi^d$. Let s_π be the stable recursion function of Σ_c/Π.

Now, for an integer $i \in \{1, \ldots, q\}$, denote by

$$(50) \quad \Lambda^i := \left\{x^{\delta(i,1)}, x^{\delta(i,2)}, \ldots, x^{\delta(i,m(i))}\right\}$$

the set of all states of Σ that appear in elements of π^i, i.e., Λ^i is the projection of π^i onto the state set X of Σ. In view of the inclusion $(\xi_0, x_0) \in \pi_0$ and the equality $\pi_0 = \pi^d$, we have $x_0 \in \Lambda^d$. The fact that Π is an equivalence partition implies that i) and ii) hold for all $i = 1, \ldots, q$.

i) $h_c((\xi^{\sigma(i,1)}, x^{\delta(i,1)})) = h_c((\xi^{\sigma(i,2)}, x^{\delta(i,2)})) = \cdots = h_c((\xi^{\sigma(i,m(i))}, x^{\delta(i,m(i))})) = h'(\zeta^i)$.

ii) For a character $u \in A$, it follows by the definition of a quotient machine that $s_\pi(\pi^i, u) = \pi^j$ if and only if the following is true: for each integer $t = 1, \ldots, m(i)$, there is an integer $\tau(t) \in \{1, \ldots, m(j)\}$ such that $s_c((\xi^{\sigma(i,t)}, x^{\delta(i,t)}), u) = (\xi^{\sigma(j,\tau(t))}, x^{\delta(j,\tau(t))})$.

Now, since $h_c((\xi, x)) = h(x)$, it follows from i) that

iii) $h(x^{\delta(i,1)}) = h(x^{\delta(i,2)}) = \cdots = h(x^{\delta(i,m(i))}) = h'(\zeta^i)$. Also, since $\pi^i \equiv \zeta^i$, it follows from ii) that, for a character $u \in A$, one has

iv) $s'(\zeta^i, u) = \zeta^j$ if and only if the following is true: for each $t = 1, \ldots, m(i)$, there exists an integer $\tau(t) \in \{1, \ldots, m(j)\}$ such that $s_c((\xi^{\sigma(i,t)}, x^{\delta(i,t)}), u) = (\xi^{\sigma(j,\tau(t))}, x^{\delta(j,\tau(t))})$.

The controller C of Fig. 1 can access the machine Σ only through the input of Σ, and the closed-loop system Σ_c operates in fundamental mode. Recalling that, in fundamental mode operation, C can drive the system Σ only along chains of stable and detectable transitions, we conclude from iv) that the following is true.

v) If $s'(\zeta^i, u) = \zeta^j$, then, for each $t = 1, \ldots m(i)$, there is a string of stable and detectable transitions that takes the controlled machine Σ from the state $x^{\delta(i,t)}$ to the state $x^{\delta(j,\tau(t))}$.

Now, consider the list of subsets $\Lambda := \{\Lambda^1, \ldots, \Lambda^q\}$, where Λ^i is given by (50), $i = 1, \ldots, q$. Since the members of Π are not empty, neither is any of the members of Λ. Consequently, iii) implies that Λ is a nondeficient subordinate list of the output equivalence list $E(\Sigma, \Sigma')$. Further, by definition of the skeleton matrix $K(\Sigma')$, the relation $s'(\zeta^i, u) = \zeta^j$ implies that $K_{ij}(\Sigma') = 1$. By (v), the same relation implies that the reachability indicator satisfies $r(\Sigma, \Lambda^i, \Lambda^j) = 1$, so that the corresponding entry of the fused skeleton matrix satisfies $\Delta_{ij}(\Sigma, \Lambda) = 1$. In other words, we have $\Delta_{ij}(\Sigma, \Lambda) = 1$ whenever $K_{ij}(\Sigma') = 1$, for all $i, j \in \{1, \ldots, q\}$. Thus, $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$; having shown earlier that $x_0 \in \Lambda^d$, our proof concludes. ♦

In Section IV-D, we provide an example of calculating the list Λ of Lemma 49. In the meanwhile, we show that the condition of Lemma 49 is not just a necessary condition but also a sufficient condition for the existence of a solution of the model matching problem. This originates from the nature of the subordinate list $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ of Lemma 49. Recall that $X' = \{\zeta^1, \ldots, \zeta^q\}$ is the state set of the model Σ'. The inequality $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ indicates the following. If the model Σ' has a stable transition from a state ζ^i to a state ζ^j, then the machine Σ has a stable and detectable transition from every state in Λ^i to a state in Λ^j. The fact that Λ is a subordinate list of $E(\Sigma, \Sigma')$ ensures that, in these transitions, the corresponding output values of the two systems match. Thus, the model matching controller needs only to generate the input string that takes Σ from a state in Λ^i to a state in Λ^j. Such a controller is constructed in the proof below, where it is also shown that the controller can be implemented as a combination of an observer and a control unit, as depicted in Fig. 2, with the observer B given by (38). We have reached one of the main results of this paper. The construction of the list Λ is described in the next subsection.

(51) Theorem: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be stably reachable asynchronous machines, where Σ' is stably minimal. Let $X' = \{\zeta^1, \ldots, \zeta^q\}$ be the state set of Σ', and assume that the initial condition of Σ' is $\zeta_0 = \zeta^d$. Then, the following two statements are equivalent.

i) There is a controller C for which $\Sigma_c = \Sigma'$, where Σ_c operates in fundamental mode and is well posed.

ii) There is a nondeficient subordinate list Λ of the output equivalence list $E(\Sigma, \Sigma')$ for which $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.

Furthermore, when ii) holds, the controller C can be designed as a combination of an observer B and a control unit F as depicted in Fig. 2, where the observer is given by (38).

Proof: The fact that i) implies ii) is stated by Lemma 49. To prove the converse direction, assume that ii) is valid. Let $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ be a subordinate list of $E(\Sigma, \Sigma')$ satisfying $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$. Using Λ, we construct a controller C for which the closed-loop system Σ_c of Fig. 1 is stably equivalent to the model Σ', is well posed, and operates in fundamental mode. The controller C we construct is a combination of an observer B and a control unit F as depicted in Fig. 2, where the observer B is given by (38). In this way, the proof will be complete upon the construction of the control unit F. Note that the control unit is an asynchronous machine $F = (A \times X, A, \Xi, \xi_0, \phi, \eta)$ with two inputs: the external input $v \in A$ of the composite system and the output $\omega \in X$ of the observer B. Our objective is to derive the recursion function ϕ and the output function η of F.

To describe the operation of the control unit F, assume that Σ' is at the stable state ζ^i and that Σ is at a stable state $\chi \in \Lambda^i$. Here, ζ^i is either the initial condition ζ^d of Σ' or the outcome of a detectable stable transition; similarly, χ is either the initial condition $x_0 \in \Lambda^d$ of Σ or the outcome of a detectable stable transition. Assume further that the external input character of Fig. 2 is switched to the character w. Then, Σ' moves to its next stable state $s'(\zeta^i, w) = \zeta^j$. Letting s be the stable recursion function of Σ, the inequality $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ implies that there is an input string $u = u_1 u_2 \ldots u_r$ such that the stable combinations $(\chi, u_1), (s(\chi, u_1), u_2), \ldots, (s(\chi, u_1 u_2 \ldots u_{r-1}), u_r)$ are all detectable, and such that the state $x_r := s(\chi, u)$ belongs to Λ^j. Define the intermediate states

$$(52) \quad x_1 := s(\chi, u_1) \quad x_2 := s(x_1, u_2), \ldots \quad x_r = s(x_{r-1}, u_r).$$

As the combinations $(x_i, u_i), i = 1, \ldots, r$, are all stable and detectable combinations, the states $x_1, \ldots, x_r$ appear as output values of the observer B immediately after having been reached by Σ. The situation can be depicted as follows.

$$\Sigma' \colon \zeta^i \xrightarrow{w} \zeta^j$$
$$\Sigma \colon \chi \in \Lambda^i \xrightarrow{u_1 u_2 \ldots u_r} x_r \in \Lambda^j.$$

The objective of the control unit F is to generate the string $u = u_1 u_2 \ldots u_r$ and apply it as input to Σ. This action achieves model matching for the present transition for the following reason. The string u drives the system Σ to the stable state x_r, which then becomes the next stable state of the closed-loop system Σ_c. Then, since $h(x_r) = h[\Lambda^j] = h'(\zeta^j)$, the next stable state of Σ_c produces the same output value as the model Σ', thus matching the model's response.

We construct now a recursion function ϕ for F that implements this behavior. Due to the requirement of fundamental mode operation, F must generate the string u one character at a time, making sure at each step that the composite system has reached a stable combination before generating the next character. As the string u has r characters, F needs r states to accomplish this, say, the states $\xi^1(\chi, \zeta^i, w), \ldots, \xi^r(\chi, \zeta^i, w)$. The resulting set of states

$$\Xi(\chi, \zeta^i, w) := \{\xi^1(\chi, \zeta^i, w), \ldots, \xi^r(\chi, \zeta^i, w)\}$$

is associated with the state ζ^i of Σ', the state χ of Σ, and the external input character w. To account for all possible such combinations, the control unit F needs the state set

$$\Xi := \xi_0 \bigcup \left\{ \bigcup_{i=1,\ldots,q} \bigcup_{\substack{\chi \in \Lambda^i \\ w \in A}} \Xi(\chi, \zeta^i, w) \right\}$$

where ξ_0 is the initial state of F. We will use the following notation. For a state x of the machine Σ, let

$$U(x) := \{a \in A : s(x, a) = x\}$$

be the set of all input characters that form stable combinations with x. Similarly, for a state ζ of the machine Σ', denote by

$$U'(\zeta) := \{a \in A : s'(\zeta, a) = \zeta\}$$

the set of all input characters that form stable combinations with ζ. Recalling that Ξ is the state set of F, the recursion function of F is a function $\phi : \Xi \times X \times A \to \Xi$, whose variables are the state $\xi \in \Xi$ of F, the output $\omega \in X$ of the observer B, and the external input character $v \in A$ of the configuration Fig. 2. Denote by $\eta : \Xi \times X \times A \to A$ the output function of F. Then, ϕ and η are defined as follows.

i) Let the closed-loop system Σ_c be at a stable combination, where Σ is at the state χ, the observer B has the output value $\omega = \chi$, and control unit F is at a state $\xi \in \Xi$. Select an element $c \in U(\chi)$, and define

$$\phi(\xi, (\chi, b)) := \xi \quad \textit{for all } b \in U'(\zeta^i)$$
$$\eta(\xi, (\chi, a)) := c \quad \textit{for all } a \in A.$$

This guaranties that, as long as the model Σ' remains at the state ζ^i, the control unit stays at the state ξ, and the system Σ stays at the state χ.

ii) Suppose that the external input switches to a character w satisfying $s'(\zeta^i, w) = \zeta^j$. As discussed earlier, the control unit F then needs to generate the input string $u = u_1 u_2 \ldots u_r$, to take Σ through the chain of states

$x_1, \ldots, x_r$ to the state $x_r \in \Lambda^j$. To this end, the recursion function ϕ must be as follows. (The output of the observer B tracks the state sequence $x_1, \ldots, x_r$.)

$$\begin{aligned}\phi(\xi, (\chi, w)) &:= \xi^1(\chi, \zeta^i, w)\\ \phi(\xi^k(\chi, \zeta^i, w), (x_k, w)) &:= \xi^{k+1}(\chi, \zeta^i, w)\\ &\qquad k = 1, 2, \ldots, r-1\\ \eta(\xi^k(\chi, \zeta^i, w), (z, b)) &:= u_k\\ \text{for } \mathit{any}\ (z, b) \in X \times A, &\qquad k = 1, 2, \ldots, r.\end{aligned}$$

iii) In response to the last input character u_r produced by F, the machine Σ reaches the desired stable state x_r. The machine Σ needs to remain at the state x_r until the external input switches from w to another character. To this end, choose an element $v \in U(x_r)$, and assign

$$\begin{aligned}\phi(\xi^r(\chi, \zeta^i, w), (x_r, w)) &:= \xi^r(\chi, \zeta^i, w)\\ \eta(\xi^r(\chi, \zeta^i, w), (z, b)) &= v \quad \mathit{for\ all}\ (z, b) \in X \times A.\end{aligned}$$

This completes the construction of the control unit F. A careful examination of this construction shows that it achieves model matching with fundamental mode operation. The fact that the output function h of Σ depends only on the state of Σ guaranties that the closed-loop configuration of Fig. 2 is well posed with the present controller. This concludes the proof. ♦

The Proof of Theorem 51 includes an algorithm for the construction of a controller C solving the model matching problem, when given a list Λ that satisfies condition ii) of the Theorem. The controller is a combination of the observer B of (38) and a state feedback controller F constructed in the proof. The derivation of an appropriate list Λ is described in Algorithm 54. In summary, the proof of Theorem 51 together with Algorithm 54 provide a complete methodology for the construction of a controller that solves the model matching problem for asynchronous sequential machines. Examples of this construction are provided in the sequel. Once an appropriate controller has been derived, its number of states can be reduced by using standard machine reduction techniques (e.g., [2]). As is the case with any asynchronous machine, the output of the closed loop machine Σ_c may include transient output values interspersed between its stable combinations. The effects of these transients on other systems can be eliminated by connecting Σ_c in series with a gate that opens only when Σ_c is in a stable combination (e.g., [2]). In effect, the output function of C built in the proof of Theorem 51 is an example of such a gate.

Two comments regarding the statement of Theorem 51 are in order. First, Theorem 51 requires the model Σ' to be a stably minimal machine. This requirement is not restrictive in any way: The model matching problem depends only on the stable-state input/output behavior of the model, and this behavior remains unaffected when the model is reduced to its stably minimal form. Second, Theorem 51 requires the system Σ to be stably reachable. Again, this is not a restrictive requirement, since, as discussed in Section 3, model matching involves only states of Σ that are stably reachable from the initial state x_0.

The statement of Theorem 51 can be simplified somewhat when the machine Σ is detectable.

(53) Corollary: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be stably reachable asynchronous machines, where Σ is detectable and Σ' is stably minimal. Assume that $h(x_0) = h'(\zeta_0)$. Then, statements i) and ii) are equivalent.

i) There is a controller C for which $\Sigma_c = \Sigma'$, where Σ_c is well posed and operates in fundamental mode.

ii) There is a nondeficient subordinate list W of the output equivalence list $E(\Sigma, \Sigma')$ such that $\Delta(\Sigma, W) \geq K(\Sigma')$. Furthermore, when ii) is valid, C can be designed as a combination of an observer and a control unit, as depicted in Fig. 2, with an observer B given by (38).

Proof: It follows directly from Theorem 51 that part i) of the Corollary implies part ii). To prove the converse, assume that $\zeta_0 = \zeta^d$. It is enough to show that when part ii) of the Corollary is valid, then one can build a subordinate list Λ that satisfies the requirements of part ii) of Theorem 51. In other words, in the notation of Theorem 51, one must show that the initial condition x_0 of Σ can be added to the member W^d of W, if it is not already there.

To this end, construct a new list $\Lambda = \{\Lambda^1, \ldots, \Lambda^q\}$ from W by setting

$$\Lambda^i = W^i, \quad \text{for all } i \neq d$$
$$\Lambda^d := \begin{cases} W^d \cup x_0 \text{ if } x_0 \notin W^d \\ W^d \text{ otherwise}\end{cases}$$

i.e., by adding the initial state x_0 of Σ to W^d if it is not already there, leaving all other members of W unchanged. We show now that the new list Λ is still a nondeficient subordinate list of $E(\Sigma, \Sigma')$ satisfying $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$.

Indeed, Λ is nondeficient since W is nondeficient. Also, since $h(x_0) = h'(\zeta_0)$ and $\zeta_0 = \zeta^d$ by assumption, and since $h[W^d] = h'(\zeta_0)$ by the definition of a subordinate list, it follows that $h[\Lambda^d] = h[W^d \cup \{x_0\}] = h[W^d] \cup h[x_0] = h'(\zeta_0) = h'(\zeta^d)$, so that $\Lambda^d \subset E^d$ (member d of the output equivalence list $E(\Sigma, \Sigma')$). As $\Lambda^j = W^j$ for all $j \neq d$ and W is a subordinate list of $E(\Sigma, \Sigma')$, we conclude that Λ is a subordinate list of $E(\Sigma, \Sigma')$. Next, since Σ is a stably reachable and detectable machine, every state of Σ is stably reachable from the initial state x_0 through a chain of stable and detectable transitions. This means that $r(\Sigma, x_0, x) = 1$ for all $x \in X$. Recalling that $\Lambda^d = W^d \cup \{x_0\}$, this implies, by Definition 41 of the reachability indicator, that

$$\begin{aligned} r(\Sigma, \Lambda^d, \Lambda^i) &= r(\Sigma, W^d, W^i) \quad \mathit{for\ all}\ i = 1, \ldots, q\\ r(\Sigma, \Lambda^i, \Lambda^d) &\geq r(\Sigma, W^i, W^d) \quad \mathit{for\ all}\ i = 1, \ldots, q.\end{aligned}$$

Adding the fact that $r(\Sigma, \Lambda^i, \Lambda^j) = r(\Sigma, W^i, W^j)$ for all $i, j \neq d$ (since $\Lambda^i = W^i$ for all $i \neq d$), we get $\Delta(\Sigma, \Lambda) \geq \Delta(\Sigma, W)$, and, since $\Delta(\Sigma, W) \geq K(\Sigma')$, we deduce that $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$. Finally, as $x_0 \in \Lambda^d$ by construction, the list Λ satisfies the requirements of Theorem 51(ii), and the proof concludes. ♦

The next objective is to devise a procedure to derive a subordinate list satisfying part ii) of Theorem 51.

D. Calculating Subordinate Lists

The following algorithm builds a list Λ that satisfies condition ii) of Theorem 51 [or condition ii) of Corollary 53], whenever such a list exists. Recall that the list Λ gives rise to a controller C that solves the model matching problem through the procedure described in the proof of Theorem 51. In this manner, the forthcoming algorithm combines with Theorem 51 into a comprehensive and constructive solution of the model matching problem. The algorithm uses a recursive process to build a decreasing chain of subordinate lists. We show later that the last list in this chain, if not deficient, satisfies condition ii) of Theorem 51; if the last list of the chain is deficient, then there is no controller that solves the requisite model matching problem.

(54) Algorithm: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be the machines of Theorem 51, let $E(\Sigma, \Sigma') = \{E^1, \dots, E^q\}$ be their output equivalence list, and let $K(\Sigma')$ be the skeleton matrix of Σ'. The following steps yield a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ \cdots \succ \Lambda(r)$ of subordinate lists of $E(\Sigma, \Sigma')$. The members of the list $\Lambda(i)$ are denoted by $\Lambda^1(i), \dots, \Lambda^q(i)$; they are subsets of the state set X of Σ.

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

Recursion Step: Assume that a subordinate list $\Lambda(k) = \{\Lambda^1(k), \dots, \Lambda^q(k)\}$ of $E(\Sigma, \Sigma')$ has been constructed for some integer $k \geq 0$. For each pair of integers $i, j \in \{1, \dots, q\}$, let $S_{ij}(k)$ be the set of all states $x \in \Lambda^i(k)$ for which $r(\Sigma, x, \Lambda^j(k)) = 0$, i.e., $S_{ij}(k)$ consists of all states $x \in \Lambda^i(k)$ for which there is no chain of stable and detectable transitions to a state of $\Lambda^j(k)$. Note that $S_{ij}(k)$ may be empty. Then, set

$$(55)\quad T_{ij}(k) := \begin{cases} S_{ij}(k) & \text{if } K_{ij}(\Sigma')m = 1 \\ \oslash & \text{if } K_{ij}(\Sigma') = 0. \end{cases}$$

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

$$(56)\quad V^i(k) := \cup_{j=1,\dots,q} T_{ij}(k), \qquad i = 1, \dots, q$$

$$(57)\quad \Lambda^i(k+1) := \Lambda^i(k) \backslash V^i(k), \qquad i = 1, \dots, q.$$

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

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

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

Algorithm 54 generates a decreasing chain of subordinate lists of the output equivalence list $E(\Sigma, \Sigma')$. Before discussing the significance of the Algorithm, we provide an example.

(58) Example: Let the machine $\Sigma = (A, Y, X, x_0, f, h)$ and the model $\Sigma' = (A, Y, X', \zeta_0, f', h')$ be as shown in Fig. 7.

(59): The corresponding stable state machines are shown in Fig. 8.

For Σ, the state set is $X = \{x^1, x^2, x^3, x^4\}$ and the initial state is $x_0 = x^1$; for Σ', the state set is $X' = \{\zeta^1, \zeta^2, \zeta^3\}$ and the initial state is $\zeta_0 = \zeta^1$. From the tables, the output equivalence list is $E(\Sigma, \Sigma') = \{E^1, E^2, E^3\}$, where $E^1 = \{x^1, x^4\}, E^2 = \{x^2\}, E^3 = \{x^3\}$. Using the tables, we obtain the fused skeleton matrix $\Delta(\Sigma)$ and the skeleton matrix $K(\Sigma')$

	a	b	c	Y
x^1	x^1	x^2	-	0
x^2	x^4	x^2	-	1
x^3	-	x^2	x^3	2
x^4	x^4	-	x^3	0

(a) The machine Σ

	a	b	c	Y
ζ^1	ζ^2	ζ^3	ζ^1	0
ζ^2	ζ^2	-	ζ^1	1
ζ^3	ζ^2	ζ^3	-	2

(b) The model Σ'

Fig. 7. Example.

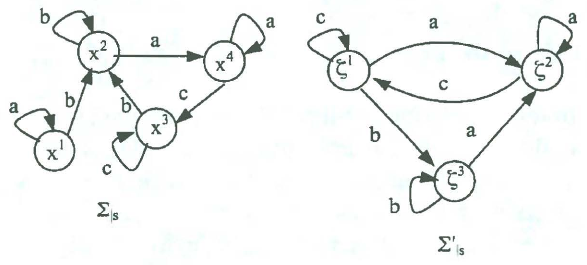


Fig. 8. Transition diagrams.

$$\Delta(\Sigma) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad K(\Sigma') = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \quad (60)$$

Algorithm 54 leads then through the following steps:

$$\Lambda(0) = E(\Sigma, \Sigma') = \{E^1, E^2, E^3\}. \quad (61)$$

At the first recursion step, set $\Lambda(1) = \{\Lambda^1(1), \Lambda^2(1), \Lambda^3(1)\}$. To find $\Lambda^1(1)$, note that $\cup_{j=1,2,3} T_{1j}(0) = \oslash$ in (56), so that $\Lambda^1(1) = \Lambda^1(0)$ by (57). Similarly, $\Lambda^2(1) = \Lambda^2(0)$, and $\Lambda^3(1) = \Lambda^3(0)$. This yields

$$(62)\quad \Lambda^1(1) = \{x^1, x^4\} \quad \Lambda^2(1) = \{x^2\} \quad \Lambda^3(1) = \{x^3\}.$$

As $\Lambda(1)$ is equal to $\Lambda(0)$, the algorithm terminates. Note that the resulting the list $\Lambda(1)$ is not deficient. ♦

We proceed now to show that the last list in the chain generated by Algorithm 54 has an important maximality property: It is the maximal subordinate list of $E(\Sigma, \Sigma')$ for which $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$.

(63) Proposition: Let Σ and Σ' be the two machines of Theorem 51, and let $E(\Sigma, \Sigma')$ be their output equivalence list. Let $\Lambda(0) \succ \Lambda(1) \succ \dots \succ \Lambda(r)$ be the chain of subordinate lists generated by Algorithm 54, and let Λ be any nondeficient subordinate list of $E(\Sigma, \Sigma')$. If $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$, then $\Lambda \prec \Lambda(r)$.

Proof: Write $E(\Sigma, \Sigma') = \{E^1, \dots, E^q\}$ and $\Lambda(k) = \{\Lambda^1(k), \dots, \Lambda^q(k)\}, k = 0, \dots, r$, and recall that $\Lambda(0) = E(\Sigma, \Sigma')$. Now, let $\Lambda = \{\Lambda^1, \dots, \Lambda^q\}$ be a subordinate list of $E(\Sigma, \Sigma')$ satisfying $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$, and assume, by contradiction, that Λ is not a subordinate list of $\Lambda(r)$. Clearly, $\Lambda \prec E(\Sigma, \Sigma')$ means that $\Lambda \prec \Lambda(0)$. Let $t \geq 0$ be the greatest integer for which $\Lambda \prec \Lambda(t)$; in view of the last two sentences, we have that $0 \leq t \leq r-1$ and that $\Lambda \not\prec \Lambda(t+1)$. The latter implies that there is an integer $i \in \{1, \dots, q\}$ such that $\Lambda^i \not\subset \Lambda^i(t+1)$. By (57), this implies that the intersection

$\Lambda^i \cap V^i(t) \neq \oslash$, so there is an element $z \in \Lambda^i \cap V^i(t)$, i.e., $z \in \Lambda^i$ and $z \in V^i(t)$. The inclusion $z \in V^i(t)$ implies, by (56), that there is an integer $j \in \{1, \ldots, q\}$ such that $z \in T_{ij}(t)$. Consequently, $T_{ij}(t)$ is not empty, and, using (55), we deduce that $r(\Sigma, z, \Lambda^j(t)) = 0$ while $K_{ij}(\Sigma') = 1$. Now, $\Lambda \prec \Lambda(t)$ entails that $\Lambda^j \subset \Lambda^j(t)$, so the second part of the last sentence implies that $r(\Sigma, z, \Lambda^j) = 0$ while $K_{ij}(\Sigma') = 1$. Finally, since $z \in \Lambda^i$, it follows that $r(\Sigma, \Lambda^i, \Lambda^j) = 0$ while $K_{ij}(\Sigma') = 1$, or that $\Delta_{ij}(\Sigma, \Lambda) = 0$ while $K_{ij}(\Sigma') = 1$, contradicting the fact that $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$. Consequently, we must have $\Lambda \prec \Lambda(r)$, and our proof concludes. ♦

The next statement indicates that the list $\Lambda(r)$ obtained at the end of Algorithm 54, if not deficient, satisfies the inequality $\Delta(\Sigma, \Lambda(r)) \geq K(\Sigma')$. This fact, when viewed together with Proposition 63, leads to an important conclusion: Algorithm 54 generates the maximal subordinate list Λ^* of $E(\Sigma, \Sigma')$ that satisfies the matrix inequality $\Delta(\Sigma, \Lambda^*) \geq K(\Sigma')$.

(64) Proposition: Let Σ and Σ' be the two machines of Theorem 51, and let $\Lambda(r)$ be the list generated by Algorithm 54. If $\Lambda(r)$ is not deficient, then $\Delta(\Sigma, \Lambda(r)) \geq K(\Sigma')$.

Proof: When $\Lambda(r)$ is not deficient, then the Test Step of Algorithm 54 implies that $r \geq 1$ and $\Lambda(r) = \Lambda(r-1)$. Assume next, by contradiction, that $\Lambda(r)$ is not deficient and $\Delta(\Sigma, \Lambda(r)) \ngeq K(\Sigma')$; since $\Lambda(r) = \Lambda(r-1)$, we can write $\Delta(\Sigma, \Lambda(r-1)) \ngeq K(\Sigma')$. There must then be a pair of integers $i, j \in \{1, \ldots, q\}$ such that $\Delta_{ij}(\Sigma, \Lambda(r-1)) = 0$ while $K_{ij}(\Sigma') = 1$. Referring to the Recursion Step of Algorithm 54, the latter implies that $S_{ij}(r-1) \neq \oslash$. But then, by (57), it follows that $\Lambda^i(r) \neq \Lambda^i(r-1)$, contradicting the equality $\Lambda(r) = \Lambda(r-1)$. Thus, we must have $\Delta(\Sigma, \Lambda(r)) \geq K(\Sigma')$, and the proof concludes. ♦

Combining Propositions 63 and 64, and noting the maximality property indicated by Proposition 63, we obtain the next result. It offers a complete characterization of the conditions for the existence of a subordinate list required for the solution of the model matching problem by way of Theorem 51.

(65) Corollary: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be stably reachable asynchronous machines, where Σ' is stably minimal. Let $X' = \{\zeta^1, \ldots, \zeta^q\}$ be the state set of Σ' and assume that the initial condition of Σ' is $\zeta_0 = \zeta^d$. Let $E(\Sigma, \Sigma')$ be the output equivalence list of Σ with respect to Σ', and let $\Lambda(r)$ be the list generated by Algorithm 54. Then, the following two statements are equivalent.

i) There is a subordinate list Λ of $E(\Sigma, \Sigma')$ for which $\Delta(\Sigma, \Lambda) \geq K(\Sigma')$ and $x_0 \in \Lambda^d$.

ii) $\Lambda(r)$ is not deficient and $x_0 \in \Lambda^d$.

In view of Corollary 65 and Theorem 51, the solution of the model matching problem can be summarized as follows.

(66) Corollary: Let $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, s', h')$ be stably reachable asynchronous machines, where Σ' is stably minimal. Let $X' = \{\zeta^1, \ldots, \zeta^q\}$ be the state set of Σ' and assume that the initial condition of Σ' is $\zeta_0 = \zeta^d$. Let $\Lambda(r) = \{\Lambda^1(r), \ldots, \Lambda^q(r)\}$ be the list generated by Algorithm 54. Then, the following two statements are equivalent.

i) There is a controller C for which $\Sigma_c = \Sigma'$, where Σ_c is well posed and operates in fundamental mode.

ii) The list $\Lambda(r)$ is not deficient and $x_0 \in \Lambda^d(r)$.

Consider again Example 59. Recall that, in this example, the outcome of Algorithm 54 was a nondeficient list. In view of Corollary 66, this implies that, for the machines Σ and Σ' of the example. there is a controller C for which the closed-loop Σ_c is stably equivalent to the model Σ', where the closed loop system operates in fundamental mode and is well posed. The controller C can be built as a combination of an observer B and a state feedback controller F. The observer B is given by (38), while the state feedback controller F is constructed from the list $\Lambda(r)$ by following the process described in the proof of Theorem 51. The construction of the controller is demonstrated in Section V. Meanwhile, we show that the computational burden of Algorithm 54 is not excessive.

(67) Proposition: Algorithm 54 has polynomial complexity.

Proof: In the notation of Algorithm 54, let n and q be the number of states of the machines Σ and Σ', respectively. A slight reflection shows that the following are valid: i) the output equivalence list $E(\Sigma, \Sigma')$ consists of q subsets, and each subset has at most n elements; ii) the chain of subordinate lists $\Lambda(0) \succ \Lambda(1) \succ \ldots \succ \Lambda(r)$ produced by Algorithm 54 strictly decreases, except, possibly, in the last step; iii) Algorithm 54 ends if a deficiency is encountered. These facts imply that the number of runs of the Algorithm's Recursion Step cannot exceed $q(n-1)$. During each such run, we check the reachability indicators $\{r(\Sigma, x, \Lambda^j(k))\}_{x \in X, j=1,\ldots,q}$. As there are no more than qn reachability indicators, the number of computations cannot exceed $q(n-1)qn < (qn)^2$, and Algorithm 54 has polynomial complexity. ♦

V. Example

Consider the asynchronous machines $\Sigma = (A, Y, X, x_0, f, h)$ and $\Sigma' = (A, Y, X', \zeta_0, f', h')$ of Example 59, where it was shown that there is a controller C solving the model matching problem $\Sigma_c = \Sigma'$. In (62), we found the subordinate list $\Lambda(1) = \{\Lambda^1(1), \Lambda^2(1), \Lambda^3(1)\}$ for which $\Delta(\Sigma, \Lambda(1)) \geq K(\Sigma')$ and $x^1 \in \Lambda^1(1)$. Using this list, we can apply the construction described in the Proof of Theorem 51 to derive the control unit $F = (A \times X, A, \Xi, \xi_0, \phi, \eta)$. This control unit is then combined with the observer B of (38), to obtain the model matching controller $C = (B, F)$ depicted in Fig. 2.

As indicated in Example 59, the initial state of Σ is x^1 and the initial state of Σ' is ζ^1. In order to keep Σ' at the state ζ^1, the external input character must be c; in order to keep Σ at the state x^1, the input character of Σ must be a. Following the Proof of Theorem 51, set the initial state of F to ξ_0. For the sake of clarity, we denote the states of the observer B by $\{x^1, x^2, x^3, x^4\}$, corresponding to the states of Σ. Thus, the initial state of B is x^1. Using the notation of the Proof of Theorem 51 and (37), it follows by Fig. 8 that $U(x^1) = \{a\}$ and $U'(\zeta^1) = \{c\}$, so that

$$\begin{aligned} F: \quad & \phi(\xi_0, (x^1, c)) := \xi_0 \\ & \eta(\xi_0, (x^1, c)) := a \\ B: \quad & \sigma(x^1, (a, \beta)) = x^1 \quad \text{for all } \beta \in Y^*. \end{aligned}$$

Assume next that the external input character switches from c to a. Recalling that s' is the stable recursion function of Σ', it follows from Fig. 8 that $s'(\zeta^1, a) = \zeta^2$. To mimic this transition, the system Σ needs to move to a state in $\Lambda^2 = \{x^2\}$, i.e., to the state x^2. We have from Fig. 8 that $s(x^1, b) = x^2$; where F needs to generate the character b to serve as input for Σ. This leads to the following:

$$\begin{aligned}
F:\quad & \phi(\xi_0, (x^1, a)) = \xi^1(x^1, \zeta^1, a)\\
& \eta(\xi^1(x^1, \zeta^1, a), (x^1, a)) = b\\
B:\quad & \sigma(x^1, (b, \beta)) = \begin{cases} x^2 \; for \; \beta = 01\\ x^1 \text{ otherwise.}\end{cases}\\
F:\quad & \phi(\xi^1(x^1, \zeta^1, a), (x^2, a)) = \xi^1(x^1, \zeta^1, a)\\
& \eta(\xi^1(x^1, \zeta^1, a), (x^2, a)) = b\\
B:\quad & \sigma(x^2, (b, \beta)) = x^2 \quad for\; all\; \beta \in Y^*.
\end{aligned}$$

Consider next the case where Σ is at a stable combination with the state $x^1 \in \Lambda^1$ and the model Σ' is at a stable combination with the state ζ^1, when the external input character switches to b. From Fig. 8, the model responds by $s'(\zeta^1, b) = \zeta^3$. To mimic this transition, the system Σ needs to move to a state in $\Lambda^3 = \{x^3\}$, i.e., to x_3. We obtain from Fig. 8 that $s(x^1, bac) = x^3$, so F needs to generate the characters b, a, and c in succession as input for Σ. This leads to the following, where, for clarity, the response of Σ is also shown:

$$\begin{aligned}
F:\quad & \phi(\xi_0, (x^1, b)) = \xi^1(x^1, \zeta^1, b)\\
& \eta(\xi^1(x^1, \zeta^1, b), (x^1, b)) = b\\
\Sigma:\quad & s(x^1, b) = x^2\\
& \beta(x^1, b) = h(x^1)h(x^2) = 01\\
B:\quad & \sigma(x^1, (b, \beta)) = \begin{cases} x^2, \text{ for } \beta = 01\\ x^1, \text{ otherwise}\end{cases}\\
F:\quad & \phi(\xi^1(x^1, \zeta^1, b), (x^2, b)) = \xi^2(x^1, \zeta^1, b)\\
& \eta(\xi^2(x^1, \zeta^1, b), (x^2, b)) = a\\
\Sigma:\quad & s(x^2, a) = x^4\\
& \beta(x^2, a) = h(x^2)h(x^4) = 10\\
B:\quad & \sigma(x^2, (a, \beta)) = \begin{cases} x^4 \text{ for } \beta = 10\\ x^2 \text{ otherwise}\end{cases}\\
F:\quad & \phi(\xi^2(x^1, \zeta^1, b), (x^4, b)) = \xi^3(x^1, \zeta^1, b)\\
& \eta(\xi^3(x^1, \zeta^1, b), (x^4, b)) = c\\
\Sigma:\quad & s(x^4, c) = x^3\\
& \beta(x^4, c) = h(x^4)h(x^3) = 02\\
B:\quad & \sigma(x^4, (c, \beta)) = \begin{cases} x^3 \text{ for } \beta = 02\\ x^4 \text{ otherwise}\end{cases}\\
F:\quad & \phi(\xi^3(x^1, \zeta^1, b), (x^3, b)) = \xi^3(x^1, \zeta^1, b)\\
& \eta(\xi^3(x^1, \zeta^1, b), (x^3, b)) = c\\
\Sigma:\quad & s(x^3, c) = x^3\\
& \beta(x^3, c) = h(x^3) = 2\\
B:\quad & \sigma(x^3, (c, \beta)) = x^3, \quad \text{for all } \beta \in Y^*.
\end{aligned}$$

Assume further that Σ is at a stable combination with the state $x^3 \in \Lambda^3$ and Σ' is at a stable combination with the state ζ^3, when the external input character switches from b to a. By Fig. 8, the model's response is $s'(\zeta^3, a) = \zeta^2$; therefore, the controller F must drive Σ to a state in $\Lambda^2 = \{x^2\}$. This is done as follows (omitting the descriptions of B and of Σ):

$$\begin{aligned}
& \phi(\xi^1(x^3, \zeta^3, a), (x^3, a)) = \xi^2(x^3, \zeta^3, a)\\
& \eta(\xi^2(x^3, \zeta^3, a), (x^3, a)) = b\\
& \phi(\xi^2(x^3, \zeta^3, a), (x^2, a)) = \xi^2(x^3, \zeta^3, a)\\
& \eta(\xi^2(x^3, \zeta^3, a), (x^2, a)) = b.
\end{aligned}$$

Next, assume that the machine Σ is in a stable combination with the state $x^2 \in \Lambda^2$ and Σ' is at the state ζ^2, when the external input character switches from a to c. This leads to the assignments

$$\begin{aligned}
& \phi(\xi^1(x^2, \zeta^2, c), (x^2, c)) = \xi^2(x^2, \zeta^2, c)\\
& \eta(\xi^1(x^2, \zeta^2, c), (x^2, c)) = b\\
& \eta(\xi^2(x^2, \zeta^2, c), (x^2, c)) = a\\
& \phi(\xi^2(x^2, \zeta^2, c), (x^4, c)) = \xi^2(x^2, \zeta^2, c)\\
& \eta(\xi^2(x^2, \zeta^2, c), (x^4, c)) = a.
\end{aligned}$$

Further, assume that Σ is at a stable combination with the state $x^4 \in \Lambda^1$ and the model Σ' is at a stable combination with the state ζ^1, when the external input character switches from c to b. Here, the necessary assignments are

$$\begin{aligned}
& \phi(\xi^1(x^4, \zeta^1, b), (x^4, b)) = \xi^2(x^4, \zeta^1, b)\\
& \eta(\xi^2(x^4, \zeta^1, b), (x^4, b)) = c\\
& \phi(\xi^2(x^4, \zeta^1, b), (x^3, b)) = \xi^2(x^4, \zeta^1, b)\\
& \eta(\xi^2(x^4, \zeta^1, b), (x^3, b)) = c.
\end{aligned}$$

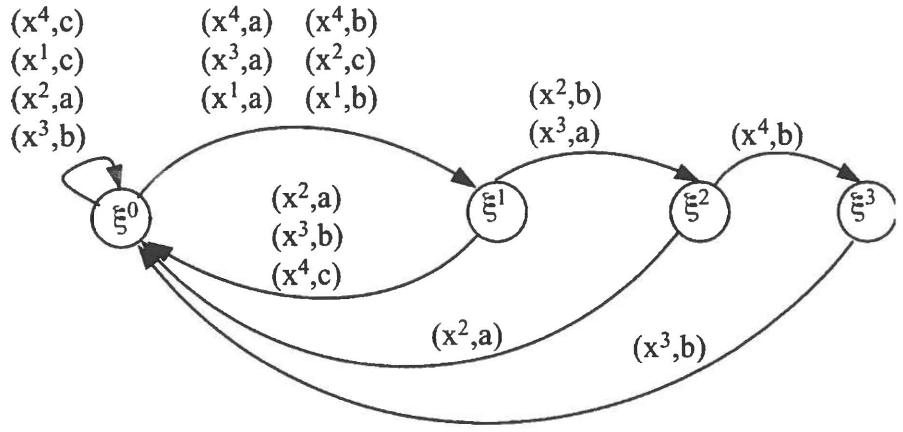
Another case: Σ is at a stable combination with the state $x^4 \in \Lambda^1$ and Σ' is at a stable combination with the state ζ^1, when the external input character switches from c to a. The corresponding assignments are

$$\begin{aligned}
& \phi(\xi^1(x^4, \zeta^1, a), (x^4, a)) = \xi^2(x^4, \zeta^1, a)\\
& \eta(\xi^2(x^4, \zeta^1, a), (x^4, a)) = c\\
& \phi(\xi^2(x^4, \zeta^1, a)(x^3, a)) = \xi^3(x^4, \zeta^1, a)\\
& \eta(\xi^3(x^4, \zeta^1, a), (x^3, a)) = b\\
& \phi(\xi^3(x^4, \zeta^1, a)(x^2, a)) = \xi^3(x^4, \zeta^1, a)\\
& \eta(\xi^3(x^4, \zeta^1, a), (x^2, a)) = b.
\end{aligned}$$

The state set of the control unit F in this case is

$$\begin{aligned}
\Xi = \{&\xi_0, \xi^1(x^1, \zeta^1, a), \xi^2(x^1, \zeta^1, a)\\
&\xi^1(x^1, \zeta^1, b), \xi^2(x^1, \zeta^1, b)\\
&\xi^3(x^1, \zeta^1, b), \xi^1(x^4, \zeta^1, b), \xi^2(x^4, \zeta^1, b)\\
&\xi^1(x^4, \zeta^1, a), \xi^2(x^4, \zeta^1, a), \xi^3(x^4, \zeta^1, a), \xi^1(x^2, \zeta^2, c)\\
&\xi^2(x^2, \zeta^2, c), \xi^1(x^3, \zeta^3, a), \xi^2(x^3, \zeta^3, a)\}.
\end{aligned}$$

An examination (e.g., [2]) shows that the state set of F can be reduced in this case to four states, with the transition diagram shown in Fig. 9.



(x^4,c) (x^1,c) (x^2,a) (x^3,b)
(x^4,a) (x^4,b) (x^3,a) (x^2,c) (x^1,a) (x^1,b)
(x^2,b) (x^3,a)
(x^4,b)
ξ^0
ξ^1
ξ^2
ξ^3
(x^2,a) (x^3,b) (x^4,c)
(x^2,a)
(x^3,b)

Fig. 9. Four-state controller.

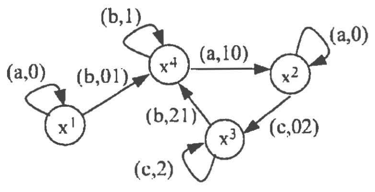


Fig. 10. Observer.

The observer $B = (A \times Y^*, X, X, x_0, \sigma, I)$ is described by Fig. 10, and is derived according to (38).

The overall controller is then obtained by combining F and B into the configuration of Fig. 2. As in this example, the number of controller states can often be reduced by using classical state reduction techniques for asynchronous sequential machines (e.g., [2]).

VI. Conclusion

The paper presents a methodology for the input/output control of asynchronous sequential machines. The methodology is based on model matching: the desired behavior is represented in the form of a model, and a controller is then designed to drive the machine so as to match the model. Necessary and sufficient conditions for the existence of a controller have been derived, and an algorithm for the construction of a controller has been presented. Whenever it exists, the controller can be implemented as a combination of an observer and a state feedback.

References

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

[2] Z. Kohavi, *Switching and Finite Automata Theory*. New York: McGraw-Hill, 1970.

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

[4] P. J. G. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," in *SIAM J. Control Optim.*, vol. 25, Jan. 1987, pp. 206–230.

[5] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," in *SIAM J. Control Optim.*, vol. 32, pp. 1075–1097.

[6] J. Hammer, "On some control problems in molecular biology," in *Proc. IEEE Conf. Decision and Control*, vol. 4, 1994, pp. 4098–4103.

[7] ——, "On the modeling and control of biological signaling chains," in *Proc. IEEE Conf. Decision and Control*, vol. 4, 1995, pp. 3747–3752.

[8] ——, "On corrective control of sequential machines," *Int. J. Control*, vol. 65, no. 65, pp. 249–276, 1996.

[9] ——, "On the control of incompletely described sequential machines," *Int. J. Control*, vol. 63, no. 6, pp. 1005–1028, 1996.

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

[11] G. Barrett and S. Lafortune, "Bisimulation, the supervisory control problem, and strong model matching for finite state machines," *J. Discrete Event Dyna. Syst.*, vol. 8, no. 4, pp. 377–429, 1998.

[12] M. D. Dibenedetto, A. Sangiovanni-Vincentelli, and T. Villa, "Model matching for finite-state machines," *IEEE Trans. Autom. Control*, vol. 46, no. 11, pp. 1726–1743, Nov. 2001.

[13] S. Eilenberg, *Automata, Languages, and Machines*. New York: Academic, 1974.

Xiaojun Geng (S'00–M'04) received the B.Sc. and M.Sc. degrees in astronautical engineering from Northwestern Polytechnic University, Xi'an, China, in 1993 and 1996, respectively, and received two Ph.D. degrees in electrical and computer engineering, one from Shanghai Jiao Tong University, Shanghai, China, in 1999, and the other from the University of Florida, Gainesville, in 2003.

Since August 2003, she has been an Assistant Professor of Electrical and Computer Engineering at California State University, Northridge. Her research interests include control of discrete-event systems, adaptive control, and evolutionary algorithms.

Jacob Hammer (M'80–S'91–F'92) received the B.Sc., M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa.

He held positions at the Technion, Case Western Reserve University, Cleveland, OH, and at the University of Florida, Gainesville, where he is currently a Professor of Electrical and Computer Engineering. His research interests are in automata theory, nonlinear control systems, and mathematical system theory.