Proceedings of the 12th Mediterranean Conference on Control and Automation, Kusadasi, Turkey, June 2004

Asynchronous Sequential Machines: Input/Output Control

Xiaojun Geng
Department of Electrical and Computer Engineering
California State University
Northridge, CA 91330, USA

Jacob Hammer
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL 32611–6130, USA

Email: hammer@mst.ufl.edu

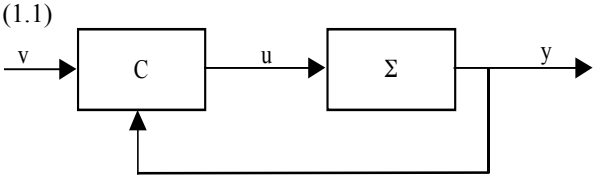
Abstract

The design of output feedback controllers for asynchronous sequential machines is considered. Main attention is devoted to the problem of model matching. Necessary and sufficient conditions for the existence of a model matching controller are presented. It is shown that, when a model matching controller exists, it can always be implemented as a combination of an observer and a state-feedback control unit. All controllers are designed so that the closed loop system operates in fundamental mode. This prevents races and hazards, and assures deterministic operation of the closed loop system.

1. Introduction

Asynchronous sequential machines are important building blocks of high-speed digital computer and control systems. The present note presents a methodology of controlling such machines and changing their behavior through the use of output-feedback control techniques.

(1.1)

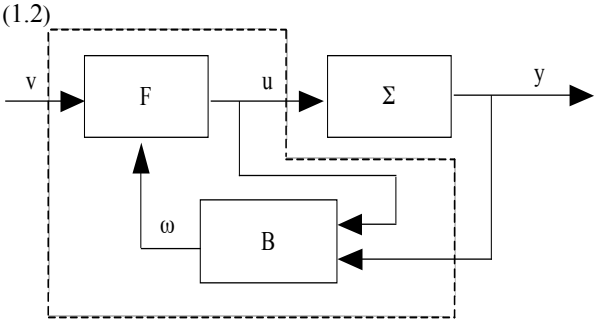


Here, Σ is the asynchronous machine being controlled and C is an asynchronous machine serving as a controller. The objective is to design C so that the closed loop system Σ_c mimics a prescribed model Σ'.

The results include necessary and sufficient conditions for the existence of an appropriate controller C. These conditions are presented in the form of a simple algorithm whose outcome determines whether or not C exists. The framework presented here builds on the results of [11] and [12], where the model matching problem is considered for machines whose state is available as output. When access to the state is not available, the problem becomes more complex and requires new analytical tools.

One outcome of the present investigation (section 3) is a general separation principle: whenever it exists, the controller C can be implemented as a combination of an *observer* B and a state-feedback *control unit* F:

(1.2)



The observer B estimates the state of Σ from input/output data of Σ; using this estimated state, the control unit F drives Σ along a desirable path.

To guaranty that a composite system of asynchronous machines is well behaved, special precautions have to be taken. Recall that an asynchronous machine has two kinds of states: stable states, i.e., states at which the machine can rest indefinitely; and unstable states - states through which the machine passes quickly as part of a transient process ([10]). If an input change occurs while the machine is in transition through a succession of unstable states, the response of the machine can become unpredictable, since the state of the machine at the time of the input change is unpredictable. One way to avoid such uncertainty is to keep the input fixed while the machine is not in a stable state. For the control loop (1.1), this requires that (i) the output of C must remain fixed while Σ is not in a stable state; and (ii) the output of Σ must remain fixed while C is not in a stable state. When these conditions are satisfied, the closed loop system *operates in fundamental mode.* All control configurations of this note operate in fundamental mode, which guarantees deterministic behavior of the closed loop systems.

Controlling asynchronous machines is significantly different from controlling other sequential machines, since one must take into consideration specialized issues related to the operation of asynchronous machines, like the distinction between stable and unstable states (section 2). There is a large body of literature about the control of other sequential machines, including [6], [7], [8], and [9], [2], [1], [3], the study of discrete event systems ([13], [14]), the references cited in these works, and more.

This note is organized as follows. Section 2 surveys terminology and background; section 3 examines the decomposition of a controller into an observer and a state-feedback control unit; and section 4 presents a solution of the model matching problem for asynchronous machines. An example is provided in section 5. Many of the topics of this note are discussed in greater detail in [5].

2. Background

An asynchronous machine is defined by a quintuple $\Sigma = (A,Y,X,x_0,f,h)$. Here, A is the input alphabet, Y is the output alphabet, X is the state set, x_0 is the initial state, $f : A\times X \to X$ is the *recursion function*, and $h : X \to Y$ is the *output function*; f and h are partial functions. The machine operates by the recursion

$$\begin{aligned} x_{k+1} &= f(x_k, u_k), \\ y_k &= h(x_k), \quad k = 0, 1, 2, \ldots , \end{aligned} \tag{2.1}$$

where $u_0, u_1, u_2, \ldots \in A$ is the *input sequence*; $y_0, y_1, y_2, \ldots \in Y$ is the *output sequence*; and $x_0, x_1, x_2, \ldots \in X$ is the *state sequence*. The step counter k advances by one upon a change of the machine's input or state. A pair $(x,u) \in X\times A$ at which the function f is defined is called a *valid pair* of Σ. When $h = I$ (the identity function), we obtain an *input/state machine*.

A valid pair $(x,u) \in X\times A$ is a *stable combination* if $x = f(x,u)$, i.e., if the machine stays indefinitely at the state x with the input u (e.g., [10]). When (x,u) is not a stable combination, the machine engages in a chain of transitions $x_1 = f(x,u)$, $x_2 = f(x_1,u)$, ... If 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 u. If there is no such i, then the machine has an *infinite cycle*. This note considers only machines without infinite cycles. Ideally, when there are no infinite cycles, it takes zero time to reach the 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 only linger at stable combinations. The behavior of stable combinations is described as follows.

Let x' be the next stable state of (x,u). The *stable recursion function* $s : X\times A \to X$ is defined by $s(x,u) := x'$ for all valid pairs (x,u). When restricted to stable combinations, the function s describes the behavior of Σ as seen by a user. The quintuple $\Sigma_{|s} := (A,Y,X,x_0,s,h)$ is the *stable state machine* induced by Σ.

Consider an input string $u = u_0u_1\ldots u_{m-1}$ 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 $x_m := s(x_0,u) := s(...s(s(s(x_0,u_0),u_1),u_2)...,u_{m-1})$ is reached.

Next, we adapt to our setting several notions of automata theory (compare to [4]).

(2.2) DEFINITION. Let $\Sigma = (A,Y,X,x_0,f,h)$ and $\Sigma' = (A,Y,X',\zeta_0,f',h')$ be machines with the same input and the same output sets. 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 machines Σ and Σ' are *stably equivalent* if $x_0 \equiv \zeta_0$. ♦

Stably equivalent machines appear identical to a user. When applied to a single machine, the presence of equivalent states indicates a redundancy.

(2.3) DEFINITION. A machine Σ is *stably reduced* if its stable state machine $\Sigma_{|s}$ has no stably equivalent states. ♦

A state x' is *stably reachable* from a state x (or x can *stably reach* x') if there is an input string u for which $x' = s(x,u)$ (see [11] and [12] for details).

(2.4) DEFINITION. A machine Σ is *stably reachable* if every state of $\Sigma_{|s}$ is stably reachable from the initial state.

The following results when all (stably) redundant and inaccessible states are removed.

(2.5) DEFINITION. An asynchronous machine is *stably minimal* if it is stably reduced and stably reachable. ♦

3. Detectability and Observers

For the control configuration (1.1) to operate in fundamental mode, it must be possible for C to determine whether Σ has reached its next stable combination. In this determination, only *input/output data* (i.e., the input string and the output string) of Σ can be used, since C has no access to the state of Σ.

(3.1) DEFINITION. An asynchronous machine Σ is *detectable* at a valid pair (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 *stable and detectable*. ♦

The following notion is critical in this context.

(3.2) DEFINITION. Let Y be an alphabet and let $y_1, ..., y_q \in Y$ be a list of characters such that $y_{i+1} \neq y_i$ for all $i = 1, ..., q-1$. Then, the *burst* of a string $y = y_1y_1 ... y_1y_2y_2 ... y_2 ... y_qy_q ... y_q$ is $\beta(y) := y_1y_2 ... y_q$. ♦

The burst is obtained by removing duplicates of neighboring characters. For example, $\beta(abbcccaa) = abca$. The burst is the only discernible entity of an asynchronous output string, since it is not possible to distinguish between equal consecutive values.

Assume that Σ is in a stable combination (x_1,v) when the input character changes to u. Let $x_1, x_2, ..., x_m$ be the string of states generated by this change, where $x_m = s(x_1,u)$ is the next stable state of Σ. If $m > 1$, set $x_{i+1} = f(x_i,u)$, $i = 1, ..., m-1$. The output string is then $h(x_1)h(x_2)\ldots h(x_m)$. The resulting burst is denoted by

$$\beta(x_1,u) := \beta(h(x_1)h(x_2)\ldots h(x_{m-1})h(x_m)). \tag{3.3}$$

To determine from input/output data whether Σ has reached the stable combination (x_m,u), one must determine whether the output string $h(x_1)h(x_2)\ldots h(x_m)$ has ended. This, however, is not always feasible. For example, when $m = 3$ and $h(x_1) = a$, $h(x_2) = b$, and $h(x_3) = b$, it is not possible to determine from the output whether the machine has reached the stable combination (x_3,u): the output switches to b at x_2, and remains unchanged during the transition from x_2 to x_3. A slight reflection shows that the end of the output string can be determined 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 leads to the next statement, which uses the notation

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

(3.4) PROPOSITION. A machine Σ is detectable at a valid pair (x,u) if and only if $\beta_{-1}(x,u) \neq \beta(x,u)$.

3.1 Skeleton Matrices.

(3.5) DEFINITION. Let Σ be an asynchronous machine with state set $X = \{x^1, ..., x^n\}$ 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

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

$i, j = 1, ..., n$. ♦

When Σ is an input/state machine, the one-step fused skeleton matrix is equal to the one-step skeleton matrix of [11] and [12]. In general, however, the two matrices may not be equal.

We review now some operations on skeleton matrices from [12]. 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 $(AB)_{ij} := \max \{A_{ik}B_{kj} : k = 1, ..., n\}$, $i, j = 1, ..., n$. With matrix combination, we can consider the k-th "power" $D^k(\Sigma)$ of $D(\Sigma)$, $k = 1, 2, ...$ Let $D^k_{ij}(\Sigma)$ be the (i,j) entry of the matrix $D^k(\Sigma)$. Define the matrix $D^{(m)}(\Sigma)$ by setting its (i,j) entry to be

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

Then, $D^{(m)}(\Sigma)$ is also a matrix of zeros and ones, and $D^{(1)}(\Sigma) = D(\Sigma)$. The case $m = n-1$, where n is the number of states of Σ, is of particular importance.

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

It can be shown that $\Delta_{ij}(\Sigma) = 1$ if and only if the state x^j can be reached from the state x^i through a chain of stable and detectable transitions ([5]). The fused skeleton matrix allows us to state simple necessary and sufficient conditions for model matching. The next notion is derived from the fused skeleton matrix.

(3.7) 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 defined as 1 if every element of Λ^1 can reach an element of Λ^2 through a chain of stable and detectable transitions; otherwise, $r(\Sigma,\Lambda^1,\Lambda^2) := 0$. ♦

To calculate the reachability indicator, consider a machine Σ with state space X and fused skeleton matrix $\Delta(\Sigma)$. Let Λ^1 and Λ^2 be two 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. Then, create a column vector V by adding all columns of $\Delta_{|\Lambda^1,\Lambda^2}(\Sigma)$. 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.

(3.8) EXAMPLE. Let $X = \{x^1, x^2, x^3\}$, $\Lambda^1 = \{x^1, x^2\}$, $\Lambda^2 = \{x^1, x^3\}$, and assume that

$$\Delta(\Sigma) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}; \text{ then}$$

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

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

3.2 Observers.

An observer is an asynchronous input/state machine whose purpose is to calculate the present state of another asynchronous machine from the input/output data of that machine. To obtain an observer for a machine $\Sigma = (A,Y,X,x_0,f,h)$, one can try to simulate the input/state part $\Sigma_f := (A,X,X,x_0,f,I)$ of Σ, where I is the identity output function. The machine Σ_f reproduces the transitions of the input/state part of Σ, but the transitions of Σ_f and Σ are not synchronized. For example, if Σ passes through a state x in response to an input string w, then Σ_f will also pass through the state x in response to w. 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. This argument shows that it is not possible to build an observer for transient states of Σ, since the machines (ideally) spend zero time in a transient state, leaving no opportunity to synchronize them. Thus, the most one can hope for is to build an observer that reveals stable combinations of Σ.

To deal with stable combinations, we focus our attention on the stable input/state machine $\Sigma_s = (A,X,X,x_0,s,I)$, where s is the stable recursion function of Σ. The next statement, which is a consequence of Definition 3.1 and Proposition 3.4, introduces an auxiliary function that helps build an observer for Σ. Denote by Y^* the set of all bursts of strings of characters of Y.

(3.9) LEMMA. Let (x_1,u) be a valid pair of the asynchronous machine $\Sigma = (A,Y,X,x_0,f,h)$ and let x_m be the next stable state of (x_1,u). When $m > 1$, define the chain of transitions $x_{i+1} = f(x_i,u)$, $i = 1, ..., m-1$, and the burst $\beta_j := \beta(h(x_1)h(x_2) \dots h(x_j))$, $j \in \{1, ..., m\}$. 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 $j = m$. ♦

Using the function κ of Lemma 3.9, we build an observer that reproduces all stable and detectable transitions of the machine Σ. The observer is an input/state machine $B = (A\times Y^*,X,X,x_0,\sigma,I)$ with two inputs: the input character $u \in A$ of Σ and the output burst $\beta \in Y^*$ of Σ. The recursion function $\sigma : X\times A\times Y^* \to X$ of B is constructed as follows. First, using the stable recursion function s of Σ, define the function $\lambda : X\times A\times\{0,1\} \to X$ by setting

$$(3.10) \quad \lambda(x,u,a) := \begin{cases} s(x,u) & \text{if } a = 1, \\ x & \text{if } a = 0. \end{cases}$$

Then, using the function κ of Lemma 3.9, set

$$\sigma(x,u,\beta_j) := \lambda(x,u,\kappa(x,u,\beta_j)).$$

The observer B uses σ as its recursion function and operates as follows. Assume that Σ is in a stable combination (x,u_{i-1}) it has reached from a detectable pair, when the input character changes to u_i, where (x,u_i) is also 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. Fundamental mode operation requires that the input character be kept constant during a chain of transitions, so $u_k = u_i$. The observer B is then the stable-state input/state machine defined by

$$(3.11) \quad B: \begin{cases} z_{k+1} = \sigma(z_k,u_k,\beta_k), \\ \omega_k = z_k, \end{cases}$$

where z_k and ω_k are the state and the output of B at the step k, respectively. A slight reflection shows that the observer displays as its output the most recent stable state the machine Σ has reached through a detectable transition. Having developed the observer B of (1.2), we turn our attention to the control unit F.

4. Controllers

4.1 Decomposition of controllers.

Recalling that the practical performance of an asynchronous machine is determined by its stable-state behavior, we formulate

(4.1) THE MODEL MATCHING PROBLEM. Given a machine Σ and a model Σ', find necessary and sufficient conditions for the existence of a controller C for which Σ_c is stably equivalent to Σ'. ♦

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, we conclude that the model Σ' can always be taken as a stably minimal machine.

Next is one of the main results of this note. It formalizes the separation principle depicted in (1.2).

(4.2) 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. Then, (ii) is valid whenever (i) is valid.

(i) There is a controller C for which $\Sigma_c = \Sigma'$, where Σ_c

is well posed and operates in fundamental mode.

(ii) The controller C can be designed as a combination of an observer and a state-feedback control unit, as depicted in (1.2), with an observer B given by (3.11).

Proof outline. The fact that Σ_c operates in fundamental mode means that all the transitions of Σ used by the closed loop system are detectable stable transitions. This implies that, at the end of each such transition, the observer B of (3.11) displays as its output the state x' of the latest stable combination reached by Σ. Now, let F be the state-feedback controller constructed in the proof of Theorem 4.3 of [12] (see also [12, Theorem 5.1]). When the state x' is fed to F, the controller F drives Σ to match the model Σ_c in fundamental mode operation. ♦

The construction of controllers that solve the model matching problem is described in [5]. Our next objective is to derive a simple algorithm for testing the solvability of the model matching problem.

4.2 Existence of controllers.

We start with terminology related to lists of subsets.

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

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 $g(\alpha) = s$. The following is a critical notion.

(4.4) 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 and the same output sets, where the state set X' of Σ' consist of the q states $\zeta^1, \ldots, \zeta^q$. Define the subsets $E^i := h^I h'(\zeta^i) \subset X$, $i = 1, ..., q$. Then, $E(\Sigma,\Sigma') := \{E^1, ..., E^q\}$ is the *output equivalence list* of Σ with respect to Σ'. ♦

In an equivalence list $E(\Sigma,\Sigma') := \{E^1, ..., E^q\}$, 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, $i = 1, ..., q$.

The following algorithm uses a recursive process to build a decreasing chain of subordinate lists. The last list of this chain determines whether the model matching problem at hand is solvable or not.

(4.5) ALGORITHM. Let $\Sigma = (A,Y,X,x_0,f,h)$ and $\Sigma' = (A,Y,X',\zeta_0,f',h')$ be the machines of Theorem 4.2, let $E(\Sigma,\Sigma') = \{E^1, \ldots, E^q\}$ be their output equivalence list, and let $K(\Sigma')$ be the skeleton matrix of Σ'. The following recursive process builds a decreasing chain $\Lambda(0) \succ \Lambda(1) \succ ... \succ \Lambda(r)$ of subordinate lists of $E(\Sigma,\Sigma')$. The members of the list $\Lambda(i)$ are denoted by $\Lambda^1(i), ..., \Lambda^q(i)$; they are subsets of the state set X of Σ.

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

Recursion Step: Assume that a subordinate list $\Lambda(k) = \{\Lambda^1(k), ..., \Lambda^q(k)\}$ of $E(\Sigma,\Sigma')$ has been constructed for some integer $k \geq 0$. For each pair of integers $i, j \in \{1, \ldots, q\}$, let $S_{ij}(k)$ be the set of all states $x \in \Lambda^i(k)$ for which the reachability indicator $r(\Sigma,x,\Lambda^j(k)) = 0$. Note that $S_{ij}(k)$ may be empty. Then, set

$$(4.6)\quad T_{ij}(k) := \begin{cases} S_{ij}(k) & \text{if } K_{ij}(\Sigma') = 1, \\ \varnothing & \text{if } K_{ij}(\Sigma') = 0 \text{ or if } S_{ij}(k) = \varnothing. \end{cases}$$

Now, define the subsets

$$(4.7)\quad V^i(k) := \bigcup_{j = 1, ..., q} T_{ij}(k),\ i = 1, ..., q.$$

Finally, using $\setminus$ to denote set difference, the next subordinate list in our decreasing chain is given by

$$(4.8)\quad \Lambda^i(k+1) := \Lambda^i(k) \setminus V^i(k),\ i = 1, ..., q.$$

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 with the value of $k+1$ as k. ♦

The next statement shows that Algorithm 4.5 provides a solution to the model matching problem. A closed loop control system is *well posed* if its output sequence is uniquely and causally determined by its input sequence and the initial conditions. Recall that fundamental mode operation guarantees a deterministic response.

(4.9) 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 and has the state set $X' = \{\zeta^1, ..., \zeta^q\}$ and the initial condition $\zeta_0 = \zeta^d$. Let $\Lambda(r) = \{\Lambda^1(r), ..., \Lambda^q(r)\}$ be the list generated by Algorithm 4.5. 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 $x_0 \in \Lambda^d(r)$.

Proof outline. Clearly, (i) means that the closed loop system Σ_c mimics the output pattern of the model Σ'. Now, all states of the set E^i generate the same output value as the state ζ^i of Σ', $i = 1, ..., q$. Thus, the model matching problem is solvable if and only if there is a subset of states of E^i that can imitate all transitions of the state ζ^i, $i = 1, ..., q$. A slight reflection shows that, recursively, the operation (4.8) removes all those states of E^i that cannot imitate some of the transitions of the state ζ^i. The remaining states form the set $\Lambda^i(r)$; they constitute

the largest subset of states of Σ that can mimic the transitions as well as the output of the state ζ^i of Σ'. Thus, the model matching problem is solvable if and only if $\Lambda^i(r) \neq \varnothing$ for all $i = 1, ..., q$, i.e., if and only if $\Lambda(r)$ is not deficient. The requirement $x_0 \in \Lambda^d(r)$ assures compatibility of the initial conditions. ♦

By Theorem 4.9, we can determine the existence of a solution to the model matching problem by using Algorithm 4.5. It is not hard to verify that Algorithm 4.5 has polynomial complexity.

5. Example

Consider two asynchronous machines $\Sigma = (A,Y,X,x_0,f,h)$ and $\Sigma' = (A,Y,X',\zeta_0,f',h')$, both having the input set $A = \{a,b,c\}$ and the output set $Y = \{0,1,2\}$. For Σ, the state set is $X = \{x^1, \ldots, x^6\}$ and the initial state is $x_0 = x^1$; for Σ', the state set is $X' = \{\zeta^1, \ldots, \zeta^4\}$ and the initial state is $\zeta_0 = \zeta^1$. The following tables indicate the corresponding recursion functions of the machines.

(5.1) The machine Σ

	a	b	c	Y
x^1	x^2	x^3	x^1	0
x^2	x^2	x^2	x^4	1
x^3	x^3	x^3	x^5	2
x^4	x^6	x^6	x^4	0
x^5	x^5	x^5	x^5	0
x^6	x^6	x^6	x^6	1

(5.2) The machine Σ'

	a	b	c	Y
ζ^1	ζ^1	ζ^2	ζ^3	0
ζ^2	ζ^2	ζ^2	ζ^4	1
ζ^3	ζ^4	ζ^4	ζ^3	2
ζ^4	ζ^4	ζ^4	ζ^4	0

We use Algorithm 4.5 to check whether there is a controller C for which Σ_c matches the model Σ'. First, the output equivalence list of Σ with respect to Σ' is

$E(\Sigma,\Sigma') = \{E^1,E^2,E^3,E^4\}$, where $E^1 = \{x^1,x^4,x^5\}$, $E^2 = \{x^2,x^6\}$, $E^3 = \{x^3\}$, and $E^4 = \{x^1,x^4,x^5\}$.

Applying Algorithm 4.5, we obtain after two recursion steps

(5.3) $\Lambda(2) = \Lambda(1) = \{\Lambda^1(2),\Lambda^2(2),\Lambda^3(2),\Lambda^4(2)\}$, where $\Lambda^1(2) = \{x^1\}$, $\Lambda^2(2) = \{x^2\}$, $\Lambda^3(2) = \{x^3\}$, and $\Lambda^4(2) = \{x^1,x^4,x^5\}$.

Since the list is not deficient, we conclude by Theorem 4.9 that there is a controller C that solves our model matching problem. For the construction of an appropriate controller, see [5]. ♦

6. References

[1]BARRETT, G., and LAFORTUNE, S., [1998], "Bisimulation, the Supervisory Control Problem, and Strong Model Matching for Finite State Machines," *Journal of Discrete Event Dynamic Systems*, Volume 8, number 4, 1998, pp. 377–429.

[2]DIBENEDETTO, M.D., SALDANHA, A., and SANGIOVANNI–VINCENTELLI, A., [1994], "Model matching for finite state machines," *Proceedings of the IEEE Conf. on Decision and Control*, vol. 3, 1994, pp. 3117–3124.

[3]DIBENEDETTO, M.D., SANGIOVANNI–VINCENTELLI, A., and VILLA, T., [2001], ″Model matching for finite–state machines,″ *IEEE Transactions on Automatic Control*, vol. 46, no. 11, 2001, pp. 1726–1743.

[4]EILENBERG, S., [1974], *"Automata, languages, and machines"*, Academic Press, NY, 1974.

[5]GENG, X., and HAMMER, J., [2003], "Input/output control of asynchronous sequential machines", submitted for publication.

[6]HAMMER, J., [1994], "On some control problems in molecular biology," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 4, 1994, pp. 4098–4103.

[7]HAMMER, J., [1995] "On the modeling and control of biological signaling chains," *Proceedings of the IEEE Conference on Decision and Control*, Vol. 4, 1995, pp. 3747–3752.

[8]HAMMER, J., [1996a] "On corrective control of sequential machines," *International Journal of Control*, Vol. 65, No. 65, 1996, pp. 249–276.

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

[10] KOHAVI, Z., [1970], "*Switching and Finite Automata Theory*," McGraw–Hill Book Company, New York, 1970.

[11] MURPHY, T.E., GENG, X., and HAMMER, J., [2002], ″Controlling races in asynchronous sequential machines,″ in *Proceedings of 2002 IFAC World Congress*, July 2002, Barcelona, Spain.

[12] MURPHY, T.E., GENG, X., and HAMMER, J., [2003] ″On the control of asynchronous machines with races,″ *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 1073-1081.

[13] RAMADGE, P.J.G., and WONHAM, W.M., [1987] "Supervisory Control of a Class of Discrete Event Processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, pp. 206 - 230, Jan. 1987.

[14] THISTLE, J. G. and WONHAM, W.M., [1994] "Control of infinite behavior of finite automata," *SIAM J. on Control and Optimization*, v 32 n 4 p 1075-1097.