European Journal of Control (2012)6:503–527
© 2012 EUCA
DOI:10.3166/EJC.18.503–527

European
Journal of
Control

Adaptive Control of Asynchronous Sequential Machines with State Feedback

Jung-Min Yang[1], Tan Xing[2], Jacob Hammer[2,*]

[1] Department of Electrical Engineering, Catholic University of Daegu, 330 Kumrak, Hayang, Gyeongsan, Gyeongbuk, 712-702, Republic of Korea;
[2] Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA

The problem of controlling deterministic asynchronous sequential machines with unknown transitions is considered. The objective is to develop adaptive state feedback controllers that acquire data about unknown transitions and utilize these data to improve closed-loop performance. Presented are adaptation and control methodologies based on an algorithm that tests unknown transitions and records their outcomes without interfering with user experience.

Keywords: Asynchronous sequential machines, adaptive control, state feedback, model uncertainty.

1. Introduction

Asynchronous sequential machines are finite state discrete systems that operate without a clock. They serve critical roles in the implementation of high speed computing systems, in parallel computing, in the modeling of signaling chains in molecular biology, and in numerous other applications. Often, the description of an asynchronous sequential machine is not fully known: the machine may not have been tested exhaustively, or its response may be affected by interferences, malfunctions, errors, or unexpected operating conditions.

For example, consider an asynchronous machine that models a signaling chain in molecular biology [5, 6]. Here, portions of the machine's behavior may be unknown due to differences among specimens or due to incompletely characterized biochemical reactions involved in the signaling chain. Another example is provided by highly complex asynchronous machines often used in computing; due to complexity, the response of such machines cannot always be verified under all excitation conditions, leaving parts of the behavior incompletely characterized. Design and implementation errors, as well as malfunctions, may also give rise to situations where a machine's response is not fully described (e.g., [15]). In these cases and in others, the underlying asynchronous machines are deterministic—their response is always the same; it is just that the outcome of some transitions is not fully characterized or known a-priori.

An *indeterminate transition* of an asynchronous machine is a deterministic transition whose outcome is not known a-priori. Once an indeterminate transition is tested and its outcome is recorded, it becomes a *determinate* transition, namely, a deterministic transition with a known outcome. A deterministic asynchronous machine that includes one or more indeterminate transitions is called an *indeterminate machine.* When attempting to control indeterminate machines, specialized algorithms must be developed to acquire data about indeterminate transitions without hindering user experience.

* *Correspondence to*: J. Hammer, E-mail: hammer@mst.ufl.edu

Received 10 May 2011; Accepted 15 December 2011
Recommended by J. Lunze, D.W. Clarke

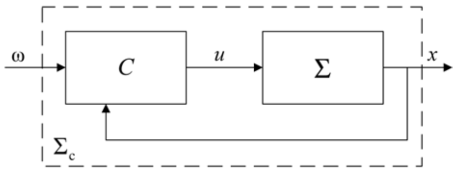

Fig. 1. Closed loop configuration.

Over time, as an indeterminate asynchronous machine is taken through its normal tasks, more and more of its indeterminate transitions may be activated and characterized. The notion of *adaptive control* refers to the process of gaining data about a machine's indeterminate transitions during operation and using these data to improve performance. An adaptive controller adjusts its functionality as it acquires more information about the controlled machine. The adaptation process, which includes data collection and controller adjustment, must be performed latently without disturbing the machine's users.

The control configuration we employ is the classical configuration described in Fig. 1. In the figure, Σ is an indeterminate asynchronous machine that must be controlled, while C is the controller—another asynchronous machine. The machine represented by the closed loop is denoted by Σ_c. The objective is to make Σ_c into a *determinate machine*, namely, into a deterministic machine with no indeterminate transitions.

The controller C collects information about Σ by recording the response of Σ to indeterminate transitions. As it gains information about indeterminate transitions of Σ, the controller C adjusts its functionality to improve control outcome. To this end, the controller C includes two functional units: one unit gathers information about the indeterminate machine Σ, while the other unit utilizes this information to control Σ. We refer to C as an *adaptive controller*. The process of gathering data about indeterminate transitions of Σ must be performed without affecting user experience and without creating uncertainty in the response of the closed loop machine Σ_c. The main control objective discussed in this report is model matching: given a determinate asynchronous machine model Σ', we develop adaptive controllers C that make the closed loop machine Σ_c indistinguishable from the model Σ'.

Before describing the principles that underlie the design of the controller C, we must review a few general features of asynchronous machines. First, recall that an asynchronous machine has two kinds of states: stable states—states in which the machine lingers until an input change occurs; and transient states—states through which the machine passes very quickly on its way from one stable state to another. Transitions through transient states are speedy, occurring ideally in zero time; as a result, transient states do not affect user experience. Nevertheless, the controller C can record transitions through transient states by using a serial register; data collected in this manner can be utilized to derive information about indeterminate transitions of Σ without interfering with user experience.

In Section 7, we develop a controller C that works along the following lines: it drives the closed loop machine Σ_c through a sequence of transient states designed to pass indeterminate transitions of Σ whose outcome is critical to achieving the performance goals of the closed loop machine Σ_c; for each indeterminate transition it activates, C records the outcome. These data are then utilized by C to improve closed loop performance. The overall effect is a process of adaptation, whereby C adjusts its action in light of the outcomes of indeterminate transitions it had recorded.

As indicated earlier, our main focus is on the development of adaptive controllers that achieve model matching. Specifically, a determinate asynchronous machine Σ' is provided as a model. The objective is to design an adaptive controller C that makes the closed loop machine Σ_c emulate Σ', notwithstanding indeterminacies of the controlled machine Σ. Explicitly, we seek an adaptive controller C for which $\Sigma_c = \Sigma'$, where the equality refers to stable transitions. Although we focus on model matching, the adaptive techniques developed here can be applied to a wide range of other design objectives.

When working with asynchronous machines, it is important to abide by *fundamental mode operation*, an operating policy that prohibits the simultaneous change of two or more variables. This policy helps prevent uncertainties: due to asynchrony, simultaneous changes in two or more variables are not possible; instead, the variables change in unpredictable sequential order, potentially leading to an unpredictable outcome.

Condition 1.1: The closed loop machine Σ_c of Fig. 1 operates in fundamental mode when all the following conditions are valid:

(i) Σ is in a stable state while C is in transition.
(ii) C is in a stable state while Σ is in transition.
(iii) The external input ω changes only when Σ and C are both in stable states. □

Parts *(i)* and *(ii)* of Condition 1.1 must be implemented during the design of the controller C; part *(iii)* on the other hand, is a restriction on the operation of the closed loop machine. Nonetheless, as transitions of asynchronous machines occur very quickly, *(iii)* does not impose a burdensome requirement.

The paper is written within the framework of [3, 4, 12, 13, 17–21], where various topics in the control of asynchronous sequential machines are considered. Studies dealing with other aspects of the control of sequential machines can be found in [11, 14, 16], where the theory of discrete event systems is investigated; in [1, 2, 5–9, 22], where issues related to control and model matching for sequential machines are studied; and in many other publications.

As indicated earlier, the present paper deals with adaptive control of asynchronous machines, paying particular attention to the process of acquiring data about indeterminate transitions without impairing user experience. This process depends in a critical manner on specialized features of asynchronous machines, such as the distinction between stable and transient states and fundamental mode operation.

The paper is organized as follows. Section 2 introduces basic features of indeterminate asynchronous machines, while Sections 3–5 discuss control theoretic features, including reachability and model matching, for such machines. Section 6 characterizes indeterminate transitions that are critical to achieving the control objective at hand, and Section 7 presents algorithms that guide the operation of adaptive controllers. The paper concludes in Section 8 with a few final remarks. A comprehensive example runs through the paper to demonstrate concepts and methodologies.

2. Indeterminate Machines

2.1. Basics

An input/state asynchronous machine Σ is represented by a quadruple $\Sigma = (A, X, x^0, f)$, where A is the input alphabet, X is the set of states, x^0 is the initial state, and $f : X \times A \to X$ is a partial function that serves as the recursion function of the machine. The machine Σ operates according to the recursion

$$x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, 2, \ldots,$$

where $x_0 := x^0$ is the initial state, and $u_0u_1u_2 \ldots$ is the input sequence; the machine generates the sequence of states $x_0x_1x_2 \ldots$ The integer k is the step counter: it advances by one upon a change of the machine's input or state. A pair $(x, u) \in X \times A$ that belongs to the domain of f is called a *valid pair.* We denote by A^* the set of all strings of characters of A and by A^+ the set of all such non-empty strings.

Often, the values of the recursion function f are not precisely known; this would be the case, for example, when Σ is afflicted by an unspecified malfunction, defect, or design error; or when the outcome of a particular transition is not well documented. The latter occurs frequently when modeling biological signaling chains, where some of the reactions may not be well characterized or may vary from specimen to specimen. In such cases, there may be several options for the machine's next state, and these can be specified as a set of potential next states. Sets of potential next states can be accommodated by regarding f as a set valued function. Letting $P(X)$ be the class of all subsets of X, consider $P(X)$ as the codomain of f, so that $f : X \times A \to P(X)$. Then, for a valid pair (x, u), the set $f(x, u)$ consists of all next state options of the machine Σ. The set $f(x, u)$ characterizes the indeterminacy of the machine at (x, u).

Definition 2.1: Let $f : X \times A \to P(X)$ be the recursion function of an asynchronous machine Σ. If $f(x, u)$ includes more than one element, then (x, u) is an *indeterminate pair of* Σ*;* in such case, (x, u) induces an *indeterminate transition* of Σ. □

Once a transition from an indeterminate pair (x, u) has occurred and its outcome x' has been recorded, the outcome of this transition will always be x'. In other words, a tested indeterminate pair (x, u) becomes a determinate pair, and the recursion function f becomes single valued at (x, u) with the value $f(x, u) := \{x'\}$. To simplify our notation, we write $f(x, u) := x'$ at pairs at which f is single valued.

If all indeterminate pairs of Σ have been tested and their outcomes acquired, the recursion function f becomes a single valued function and Σ becomes a determinate machine. It is not always necessary to test all indeterminate pairs of a machine—it suffices to test pairs that are involved in the machine's tasks.

A valid pair (x, u) of the machine Σ is a *stable combination* if $f(x, u) = x$, namely, if x is a fixed point of the function f with the input u; otherwise, (x, u) is a *transient combination*. At a stable combination, the machine Σ lingers until an input change occurs; on the other hand, Σ passes quickly through transient combinations (ideally, in zero time). A transition of Σ from one stable combination to another may take the machine through a chain of several transient combinations. Specifically, assume that Σ is at a stable combination (x, u'), when the input character changes to u. This may result in a chain of transitions $x_1 := f(x, u)$, $x_2 := f(x_1, u)$, … If this chain terminates, then there is an integer $i \geq 1$ for which $x_i = f(x_i, u)$, and x_i is the *next stable state* of x with the input u. If the chain does not terminate, then Σ has an infinite cycle. In the present paper, we restrict our attention to machines with no infinite cycles.

Convention 2.2: *Only machines without infinite cycles are considered.* □

Thus, for machines considered in this paper, every valid combination has a next stable state. The *stable recursion* *function* $s : X \times A \to P(X)$ of the machine Σ is defined at every valid pair (x, u) of Σ by $s(x, u) := \{x'\}$, where x' is the next stable state (or the set of potential next stable states) of x with the input u. As before, we shorten our notation by writing $s(x, u) := x'$ for a determinate stable transition.

The stable recursion function gives rise to the *stable state machine* $\Sigma_{|s} = (A, X, x^0, s)$, which represents the stable state behavior of Σ; the machine $\Sigma_{|s}$ is also an input/state machine [4, 13]. When $s = f$, then Σ is itself a stable state machine.

As transient states are traversed very quickly (ideally, in zero time), they do not affect the experience of a machine's users. Users are aware only of stable combinations, since these are the only combinations at which a machine may linger. Consequently, states that have no stable combinations will never be noticed by a user and can be ignored. In this regard, it is common practice to include in the state set only states that have a stable combination. We adhere to this practice: in our notation, every member of the state set X of the controlled machine Σ has at least one stable combination.

It is important to note that transient combinations do play a critical role in feedback control of asynchronous machines. Indeed, the controller C of Fig. 1 works by turning undesirable stable combinations of Σ into transient combinations of the closed loop machine Σ_c. This dismisses undesirable features of the response of Σ and molds the closed loop response to the desired model Σ' (see [4, 13]). Furthermore, as discussed in Section 7, transients of the closed loop machine Σ_c are employed by the controller C to test indeterminate transitions of Σ and acquire their outcomes. This forms a vital component of the adaptive control process.

For a string of input characters $u := u_0 u_1 \cdots u_q$, it is convenient to use the shorthand notation

$$s(x, u) := s(s(\ldots s(s(s(x, u_0), u_1), u_2) \ldots), u_q)$$

to denote the final stable state that Σ reaches when the input string u is applied starting at the state x. To preserve fundamental mode operation, the string u must be applied character-by-character, waiting after each input character for Σ to reach its next stable state before applying the next input character. A state x' is *stably reachable* from a state x if there is an input string u for which $x' \in s(x, u)$.

Finally, note that, in an asynchronous environment, it is impossible to distinguish between consecutive identical characters of a string, since there is no specified time duration to signify the end of one instance and the beginning of the next instance of the same character. Thus, a string of characters such as *aabbbcc* is indistinguishable from the string *abc*. This fact is used tacitly in our discussion.

2.2. Indeterminate Transitions

Consider an indeterminate input/state asynchronous machine $\Sigma = (A, X, x^0, f)$ with the stable recursion function s. If s has no indeterminate pairs, then Σ behaves like a determinate machine for all practical purposes, since only stable transitions matter for user experience. In other words, indeterminate transitions have an impact on user experience only when they affect the stable recursion function s. Let the indeterminate pairs of s be given by the set

$$U := \{(z^1, u^1), (z^2, u^2), \ldots, (z^r, u^r)\} \subseteq X \times A. \quad (2.1)$$

The next stable state of a pair $(z^i, u^i) \in U$ is not precisely known; instead, a subset

$$Z^i := \{z^{i,1}, \ldots, z^{i,n(i)}\} \subseteq X \quad (2.2)$$

of $n(i) > 1$ states is specified as the set of potential next stable states, so that

$$s(z^i, u^i) := Z^i, i = 1, 2, \ldots, r.$$

The set Z^i is called the set of *potential outcomes* of (z^i, u^i).

Recall that an indeterminate machine Σ is deterministic – its response is always the same; it is just that the response to indeterminate pairs is not precisely known a-priori. Once the next stable state $z' \in Z^i$ of an indeterminate pair (z^i, u^i) is found experimentally, the pair (z^i, u^i) becomes determinate: the machine Σ will always move to z' in response to (z^i, u^i). An important task of the controller C of Fig. 1 is to test and collect data about outcomes of indeterminate pairs and utilize these data in future action.

To help us work more efficiently with indeterminate transitions, we introduce a convenient algebraic framework. First, we resolve formally the indeterminacy about the outcome of an indeterminate pair (z^i, u^i) by associating a unique pseudo input character with every potential outcome, as follows. Let B be an alphabet disjoint from the input alphabet A of Σ and containing at least $n(1) + n(2) + \cdots + n(r)$ characters. For each indeterminate pair (z^i, u^i) of the stable recursion function s of Σ, choose a distinct set A^i of $n(i)$ characters of B, say

$$\begin{aligned} &A^i := \{u^{i,1}, u^{i,2}, \ldots, u^{i,n(i)}\} \subseteq B, \text{ where} \\ &A^i \cap A^j = \varnothing \quad \text{for all } i \neq j \in \{1, \ldots, r\}. \end{aligned} \quad (2.3)$$

We refer to A^i as the *pseudo alphabet* associated with the indeterminate pair (z^i, u^i). Denote by

$$A' := \bigcup_{i=1,\ldots,r} A^i \quad (2.4)$$

the set of all pseudo characters and define the *extended input alphabet*

$$\tilde{A} := A \bigcup A'. \tag{2.5}$$

Now, construct from Σ a determinate asynchronous machine by associating a distinct pseudo input character with every potential outcome of an indeterminate transition, as follows.

Definition 2.3: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with the stable recursion function s, the set of indeterminate pairs U of (2.1), the potential outcomes of (2.2), and the extended alphabet $\tilde{A}$ of (2.5). Then, the *adjusted stable recursion function* $s_a : X \times \tilde{A} \to X$ of Σ is

$$s_a(x,u) :=$$

$$\begin{cases} s(x,u) & \text{if } (x,u) \text{ is a determinate pair of } \Sigma; \\ z^{i,j} & \text{if } (x,u) = (z^i, u^{i,j}) \text{ for some} \\ & \quad i \in \{1,\ldots,r\} \text{ and } j \in \{1,2,\ldots,n(i)\}; \\ z^{i,j} & \text{if } (x,u) = (z^{i,j}, u^{i,j}) \text{ for some} \\ & \quad i \in \{1,\ldots,r\} \text{ and } j \in \{1,2,\ldots,n(i)\}; \\ \text{invalid} & \text{for all other } (x,u) \in X \times \tilde{A}, \\ & \quad \text{including all } (x,u) \in U. \end{cases} \tag{2.6}$$

The asynchronous machine $\Sigma_a := (\tilde{A}, X, x^0, s_a)$ is called the *adjusted machine* of Σ. A set of strings $\gamma \subseteq (\tilde{A})^+$ is *valid* at a state $x \in X$ if $s_a(x,t)$ is valid for all strings $t \in \gamma$. □

In (2.6), the first line makes Σ_a identical to Σ at determinate pairs of Σ; the second line formally resolves the indeterminacy of each indeterminate pair (z^i, u^i) by associating a distinct pseudo input character $u^{i,j} \in \tilde{A}$ with every potential outcome $z^{i,j}$; the third line makes each pair $(z^{i,j}, u^{i,j})$ into a stable combination of s_a; and the fourth line turns all members of the set U of indeterminate pairs into invalid pairs of s_a. Then, the resulting adjusted machine Σ_a has no indeterminate pairs; it is a determinate and stable input/state asynchronous machine.

Example 2.4: Consider the asynchronous machine Σ with the input alphabet $A = \{a,b,c,d\}$, the state set $X = \{x^1, x^2, x^3, x^4\}$, the initial state $x^0 = x^1$, and the stable recursion function s described by Fig. 2 and Table 1; in the table, "$-$" denotes an invalid pair.

For the present machine, the pairs (x^2, c) and (x^3, a) are indeterminate; hence,

$$U = \{(x^2, c), (x^3, a)\}.$$

To obtain the adjusted machine, we introduce the pseudo character sets

$$A^1 := \{c^1, c^2\} \text{ and } A^2 := \{a^1, a^2\}.$$

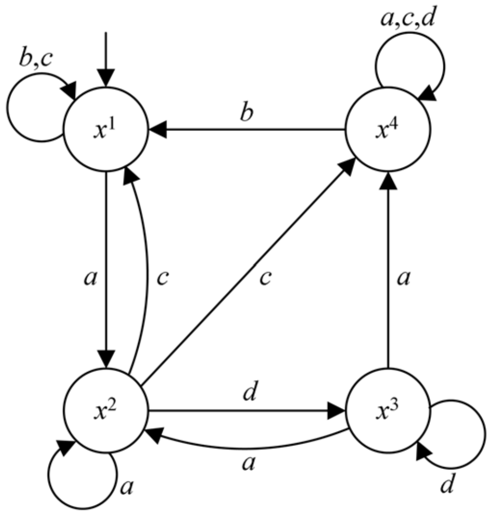

Fig. 2. The machine Σ.

Table 1. The stable recursion function of Σ.

State	a	b	c	d
x^1	x^2	x^1	x^1	$-$
x^2	x^2	$-$	$\{x^1, x^4\}$	x^3
x^3	$\{x^2, x^4\}$	$-$	$-$	x^3
x^4	x^4	x^1	x^4	x^4

Table 2. The adjusted stable recursion function.

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

In view of (2.4) and (2.5), these yield the pseudo input alphabet

$$A' = A^1 \cup A^2 = \{a^1, a^2, c^1, c^2\}$$

and the extended input alphabet

$$\tilde{A} = \{a,b,c,d,a^1,a^2,c^1,c^2\}.$$

By (2.6), the adjusted stable recursion function s_a of Σ is described in Table 2.

The adjusted stable recursion function s_a determines the adjusted machine Σ_a, whose transition map is depicted in Fig. 3; as we can see, Σ_a has no indeterminate transitions. □

The adjusted machine Σ_a is a determinate 'synthetic' machine: each valid pair of Σ_a has a single next stable

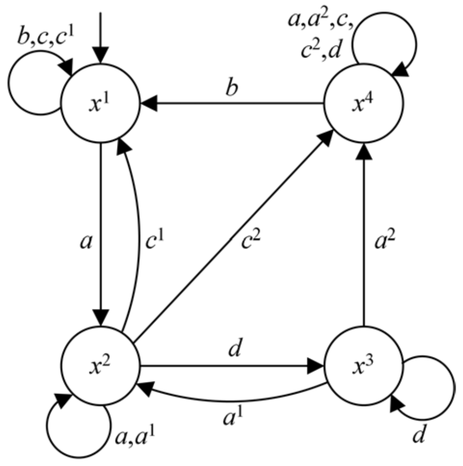

Fig. 3. State flow diagram of Σ_a.

state. Of course, members of the pseudo character set A' cannot be applied as inputs to the real machine Σ. Nevertheless, as will become clear from our ensuing discussion, they serve a critical role in helping derive adaptive control algorithms for Σ.

3. A Semiring and Complete Sets

3.1. Complete Sets

To work with the extended alphabet $\tilde{A}$, we introduce a semiring $\mathfrak{A}$ over the set $(\tilde{A})^*$ of all strings of characters of $\tilde{A}$. In this semiring, concatenation serves as multiplication and union serves as addition. Explicitly, the product of two strings $a, b \in (\tilde{A})^*$ is the concatenated string ab, where a is the prefix (the earlier part) and b is the suffix (the later part) of the result; clearly, this multiplication is not commutative. The sum of two subsets of strings $c, d \subseteq (\tilde{A})^*$ is the set of strings formed by their union, i.e., $c + d := c \cup d$. For expressions involving both operations, we introduce distributive laws: given three strings $a, b, c \in (\tilde{A})^*$, define

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

A direct verification shows that, under these operations, the set of strings $(\tilde{A})^*$ becomes a semiring, with the empty string $\varnothing$ serving as identity for both operations.

To examine the significance of the semiring $\mathfrak{A}$, consider an indeterminate pair (z^i, u^i) of an asynchronous machine Σ with the set of potential outcomes $\{z^{i,1}, \ldots, z^{i,n(i)}\}$, and let $A^i = \{u^{i,1}, \ldots, u^{i,n(i)}\}$ be the corresponding set of pseudo input characters. Then, by (2.6), the adjusted stable recursion function satisfies $z^{i,j} = s_a(z^i, u^{i,j})$ and $z^{i,j} = s_a(z^{i,j}, u^{i,j})$, $j = 1, \ldots, n(i)$, $i = 1, \ldots, r$. Now, assume that Σ has a state z' with the following feature: for each one of the outcomes $z^{i,j}$, there is an input string $\alpha^j \in A^*$ (with no pseudo characters) for which

$$s(z^{i,j}, \alpha^j) = z', \; j = 1, \ldots, n(i).$$

Then, irrespective of which one of the states $z^{i,1}, \ldots, z^{i,n(i)}$ is the actual outcome of (z^i, u^i), we can always reach the state z' by using a state feedback controller: upon detecting the outcome $z^{i,j}$, the controller applies to Σ the input string α^j. This takes Σ to the state z' for every possible outcome of the indeterminate transition.

More generally, for the adjusted machine Σ_a, every set of strings of the form

$$\gamma^1(i) := \left\{ u^{i,1}\alpha^1 + u^{i,2}\alpha^2 + \cdots + u^{i,n(i)}\alpha^{n(i)} \;\middle|\; \begin{array}{l} \alpha^1, \alpha^2, \ldots, \alpha^{n(i)} \in A^* \text{ and} \\ s_a(z^i, u^{i,1}\alpha^1) = s_a(z^i, u^{i,2}\alpha^2) = \cdots = s_a(z^i, u^{i,n(i)}\alpha^{n(i)}) \end{array} \right\} \tag{3.1}$$

gives rise to a determinate stable transition from the indeterminate pair (z^i, u^i) to a common stable state.

Further, assume that the machine Σ started at a determinate pair (z, u) and was taken by an input string $\alpha \in A^*$ to the indeterminate pair (z^i, u^i). Then, using α as a prefix, we obtain the set of strings

$$\Gamma^1(i) := \left\{ \alpha\gamma^1(i) \;\middle|\; \begin{array}{l} \alpha \in A^* \text{ takes } \Sigma \text{ from a determinate} \\ \text{pair to the indeterminate pair } (z^i, u^i), \\ \text{and } \gamma^1(i) \text{ is given by (3.1).} \end{array} \right\} \tag{3.2}$$

The set of strings $\Gamma^1(i)$ includes a response to every possible outcome of the indeterminate pair (z^i, u^i), and, as a whole, induces the same end result in all cases. Note that $\Gamma^1(i)$ has two critical features:

(a) It includes a reaction to every possible outcome of the indeterminate pair (z^i, u^i); and
(b) All these reactions ultimately take Σ to the same stable state.

We refer to a set of the form $\Gamma^1(i)$ as a *complete set of order* 1; the integer i indicates that it is associated with the indeterminate pair (z^i, u^i). Assuming that Σ has the r indeterminate pairs $\{(z^1, u^1), \ldots, (z^r, u^r)\}$ and collecting all the associated complete sets of order 1, we obtain the family of complete sets of order 1 of Σ

$$\Gamma^1(\Sigma) := \bigcup_{i=1,\ldots,r} \Gamma^1(i).$$

A brief examination shows that features *(a)* and *(b)* remain valid when the strings $\alpha^1, \alpha^2, \ldots, \alpha^{n(i)}$ of (3.1) include complete sets of order 1, i.e., when $\alpha, \alpha^1, \alpha^2, \ldots, \alpha^{n(i)} \in (A \cup \Gamma^1(\Sigma))^*$. Substituting into (3.2), this creates the family $\Gamma^2(i)$ of complete sets of order 2 associated with the indeterminate pair (z^i, u^i). As before, the entire family of complete sets of order 2 of Σ is then

$$\Gamma^2(\Sigma) := \bigcup_{i=1,\ldots,r} \Gamma^2(i).$$

It is convenient to define

$$\Gamma^0(\Sigma) := \{\alpha \in A^* | \alpha \text{ takes } \Sigma \text{ along a path that includes no indeterminate pairs}\}. \quad (3.3)$$

In general, assuming that the family $\Gamma^p(\Sigma)$ of complete sets of order p of Σ has been created for an integer $p \geq 1$, define the set $\gamma^{p+1}(i)$ by

$$\gamma^{p+1}(i) := \left\{ u^{i,1}\alpha^1 + u^{i,2}\alpha^2 + \cdots + u^{i,n(i)}\alpha^{n(i)} \,\middle|\, \begin{array}{l} \alpha^1, \alpha^2, \ldots, \alpha^{n(i)} \in \left[\left(\cup_{j=0,\ldots,p}\Gamma^j(\Sigma)\right)\right]^* \text{ and} \\ s_a(z^i, u^{i,1}\alpha^1) = s_a(z^i, u^{i,2}\alpha^2) = \cdots = s_a(z^i, u^{i,n(i)}\alpha^{n(i)}) \end{array} \right\}. \quad (3.4)$$

Then, family $\Gamma^{p+1}(i)$ of all complete sets of order $p+1$ associated with the indeterminate pair (z^i, u^i) is

$$\Gamma^{p+1}(i) := \left\{ \alpha\gamma^{p+1}(i) \,\middle|\, \begin{array}{l} \alpha \in \left[\left(\cup_{j=0,\ldots,p}\Gamma^j(\Sigma)\right)\right]^* \text{ takes} \\ \Sigma \text{ from a determinate pair to} \\ \text{the indeterminate pair } (z^i, u^i), \\ \text{and } \gamma^{p+1}(i) \text{ is given by (3.4).} \end{array} \right\} \quad (3.5)$$

Finally, the family of all complete sets of order $p+1$ of Σ is

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

Definition 3.1: A *complete set* of Σ is any member of the family

$$\Gamma(\Sigma) := \bigcup_{p=0,1,2,\ldots} \Gamma^p(\Sigma). \square$$

A complete set includes a response to every possible outcome of every indeterminate transition encountered along its way, and all these responses ultimately lead the machine to the same stable state. Note that each complete set of Σ is associated with a specific valid pair of the adjusted machine Σ_a at which the input sequences of the complete set start to act. This can be a determinate or an indeterminate pair.

The following notation is convenient. For a set of states $Y \subseteq X$ and a set of strings $\gamma \subseteq (\tilde{A})^*$, denote

$$s_a(Y, \gamma) := \left\{ s_a(x, u) \,\middle|\, \begin{array}{l} x \in Y, u \in \gamma, \text{ and} \\ (x, u) \text{ is a valid pair.} \end{array} \right\}.$$

The significance of complete sets originates from the fact that they are intimately related to the existence of state feedback controllers that induce determinate transitions in an indeterminate machine, as follows.

Theorem 3.2: *Let Σ be an indeterminate machine with the adjusted machine $\Sigma_a = (\tilde{A}, X, x^0, s_a)$, and let x and x' be two states of Σ. Then, the following two statements are equivalent.*

(i) There is a state feedback controller that takes Σ through a determinate transition from x to x' in fundamental mode operation.
(ii) There is a complete set γ satisfying $s_a(x, \gamma) = x'$.

Proof: Assume first that *(i)* is valid, and let C be a state feedback controller for which the closed loop machine Σ_c has a determinate transition from x to x' in fundamental mode operation. Then, C generates a response for every outcome of every indeterminate transition of Σ encountered along the way from x to x', and each of these responses takes Σ to x'. Translating all responses of C to actions on the adjusted machine Σ_a, let $\gamma \subseteq (\tilde{A})^+$ be the set of all extended alphabet strings generated by C while taking Σ_a from x to x'. We have to show that γ is a complete set. Let p be the maximal number of indeterminate transitions that Σ encounters as it is guided by C from x to x'. We prove *(ii)* by induction on p.

First, for $p = 0$, the controller C generates an input string $\gamma \in A^*$ that takes Σ_a from x to x' without encountering any indeterminate transitions of Σ, so we have $\gamma \in \Gamma^0(\Sigma)$ by (3.3). Hence, γ is a complete set when $p = 0$. Next, assume that γ is a complete set for an integer $p = q \geq 0$, and consider the case $p = q + 1$.

Let $(z^i, u^i) \in X \times A$ be the first indeterminate pair that Σ encounters as it is being taken by C from x to x', and let $\alpha \in A^+$ be the string that takes Σ from x to z^i. Also, let $\{z^{i,1}, \ldots, z^{i,n(i)}\}$ be the set of potential next stable states of (z^i, u^i), and let $u^{i,j}$ be the pseudo character that leads Σ_a from the state z^i to the state $z^{i,j}$. Then, C must generate a response $\alpha^{i,j} \subseteq (\tilde{A})^+$ to each outcome $z^{i,j}$, and this response induces a determinate transition from $z^{i,j}$ to x', $j = 1, 2, \ldots, n(i)$. Along this path, Σ cannot encounter more than q indeterminate pairs, since the pair (z^i, u^i) has already been encountered and is therefore excluded from the count. Thus, by our induction assumption, $\alpha^{i,j} \subseteq \Gamma^q(\Sigma)$ for all $j = 1, 2, \ldots, n(i)$. Summarizing, we have $\gamma = \alpha(u^{i,1}\alpha^{i,1} + u^{i,2}\alpha^{i,2} + \cdots + u^{i,n(i)}\alpha^{i,n(i)})$ with $\alpha^{i,j} \subseteq \Gamma^q(\Sigma)$ for all $j = 1, 2, \ldots, n(i)$; by (3.5), this shows that $\gamma \in \Gamma^{q+1}(\Sigma)$, and *(i)* implies *(ii)* by induction.

Conversely, assume that *(ii)* is valid, and let γ be a complete set of Σ. We use γ to construct a controller C that induces a determinate transition from x to x'. To this end, assume that a controller implementing the first k steps of a member of γ has been built. If this brings Σ to a determinate pair, then, in step $k+1$, the controller applies to Σ the next character of the string. If Σ encounters an indeterminate pair (z^i, u^i) at step k, then, in view of (3.4) and (3.5), the complete set γ prescribes the next input character $u^{i,j}$ that must be applied to Σ at each outcome $z^{i,j}$ of (z^i, u^i). The construction of [13, Proof of Theorem 4.3] allows us to build a controller that generates the character $u^{i,j}$ when detecting the state $z^{i,j}$. Continuing this construction step-by-step as in the reference, we obtain from γ a controller C that drives Σ through a determinate transition from x to x'. This concludes our proof. □

The *length* $|\gamma|$ of a set of strings $\gamma \subseteq (\tilde{A})^+$ is the number of characters in the longest member of γ.

Proposition 3.3: *Let $\Sigma_a = (X, \tilde{A}, x^0, s_a)$ be an adjusted machine with n states. If there is a complete set γ that takes Σ_a from a stable combination with a state x to a stable combination with a state x', then there also is such a complete set γ' of length $|\gamma'| \le n-1$.*

Proof: Let A be the original input alphabet of the machine Σ, let A' be the pseudo input alphabet of Σ_a, and let $u = u_1u_2\ldots u_{|u|} \in (\tilde{A})^*$ be a member of γ of length $|u| \ge n$. While Σ_a is driven by u from a stable combination with the state x to a stable combination with the state x', it encounters $n+1$ or more states, say the states $x_0, x_1, x_2, \ldots, x_{|u|}$, where $x_0 := x$, $x_{|u|} := x'$, and $x_{i+1} = s_a(x_i, u_{i+1})$, $i = 0, 1, \ldots, |u|-1$. Consider the combinations (x_0, u_1), $(x_1, u_2), \ldots, (x_{|u|-1}, u_{|u|})$, $(x_{|u|}, u_{|u|})$ encountered along the way, where the last pair is the stable combination at the end state x'. Note that, except for the last pair, all pairs in this list are transient combinations leading to the next stable state of Σ_a: each pair (x_i, u_{i+1}), $i = 0, 1, \ldots, |u|-1$, comes after a stable combination (x_i, u_i) of Σ_a and leads to the next stable combination (x_{i+1}, u_{i+1}) of Σ_a. Note also that changing the input character u_{i+1} in the pair (x_i, u_{i+1}) to another input character will not compromise fundamental mode operation, since this input character is applied while Σ_a is in its previous stable combination (x_i, u_i). Such a string of pairs is generated for each member of γ; let $\gamma^\times \subseteq (X \times \tilde{A})^*$ be the set of all such strings of state/input pairs generated when strings of γ are applied to Σ_a as inputs.

Now, considering that Σ_a has only n states, the string of pairs $(x_0, u_1), (x_1, u_2), \ldots, (x_{|u|-1}, u_{|u|}), (x_{|u|}, u_{|u|})$ must include two steps $i \ne j$, $0 \le i < j \le |u|$, at which $x_i = x_j$. Denote $\xi := x_i = x_j$, $u' := u_i$, and $u'' := u_j$. We distinguish now between the following cases.

Case 1. $(\xi, u'), (\xi, u'') \in X \times A$, i.e., (ξ, u') and (ξ, u'') are both determinate pairs of Σ:
Denote by $\gamma((\xi, u'), (\xi, u''))$ the subset of $\gamma^\times$ that consists of all strings of state/input pairs in which the pair (ξ, u') appears strictly before the pair (ξ, u'').

Consider a typical member $v = v_1v_2\ldots v_{|v|} \in \gamma((\xi, u'), (\xi, u''))$. Denote by $k_1(v)$ the step number at which the pair (ξ, u') appears in v, and by $k_2(v)$ the step number at which the pair (ξ, u'') appears in v. Then, by our selection of the members of $\gamma((\xi, u'), (\xi, u''))$, we have $k_2(v) > k_1(v)$. The precise values of $k_1(v)$ and $k_2(v)$ may vary from one member of $\gamma((\xi, u'), (\xi, u''))$ to another. Now, replace every member $v \in \gamma((\xi, u'), (\xi, u''))$ by the shortened string $v' := v_1v_2\ldots v_{k_1(v)-1}v_{k_2(v)}\ldots v_{|v|}$ obtained by excising the segment $v_{k_1(v)}, v_{k_1(v)+1}, \ldots, v_{k_2(v)-1}$ from v; here, $v_{k_1(v)-1} := \varnothing$ when $k_1(v) = 1$. Denote by $\gamma_1^\times$ the resulting set of strings of state/input pairs, and let γ_1 be the corresponding set of input strings. By construction, some strings of γ_1 are strictly shorter than the strings of γ from which they originate. As no indeterminate pairs were modified in this process, it follows that γ_1 is still a complete set.

Case 2. $(\xi, u') \in X \times A'$, namely, u' is a pseudo input character, while $(\xi, u'') \in X \times A$, i.e., u'' is an original input character:
Denote by $A'(\xi, u')$ the set of all pseudo characters associated with the indeterminate transition of which (ξ, u') represents one option. Now, consider a member $v = v_1v_2\ldots v_{|v|} \in \gamma^\times$; denote by $k_1(v)$ the position of the pair (ξ, u') in v, and by $k_2(v)$ the position of the pair (ξ, u'') in v, where $k_2(v) > k_1(v)$. As γ is a complete set, $\gamma^\times$ must include, for every member $u \in A'(\xi, u')$, a string that starts with the same $k_1(v) - 1$ terms as v and has the pair (ξ, u) in position $k_1(v)$; that is, a string of the form $v(u) = v_1v_2\cdots v_{k_1(v)-1}(\xi, u)v_{k_1(v)+1}(u)\cdots$ Excising the segment $(\xi, u)v_{k_1(v)+1}(u)\cdots v_{k_2(v(u))-1}(u)$, replace each string of the form $v(u)$ in $\gamma^\times$ by

the string $v'(u) := v_1 v_2 \ldots v_{k_1(v)-1} v_{k_2(v(u))}(u) \ldots v_{|v(u)|}(u)$. Repeat this process for every $u \in A'(\xi, u')$. Denote by $\gamma_1^\times$ the resulting set of strings of pairs, and let γ_1 be the corresponding set of input strings. Note that every string $v(u)$ was shortened by this process, while no strings were made longer. Since all remaining indeterminate transitions in $\gamma_1^\times$ are identical to their counterparts in $\gamma^\times$, it follows that γ_1 is still a complete set.

Case 3. $(\xi, u') \in X \times A$, while $(\xi, u'') \in X \times A'$, i.e., u' is an original input character, while u'' is a pseudo input character:
Let $A'(\xi, u'')$ be the set of all pseudo characters associated with the indeterminate transition of which (ξ, u'') represents one option. Now, consider a member $v = v_1 v_2 \ldots v_{|v|} \in \gamma^\times$; denote by $k_1(v)$ the position of the pair (ξ, u') in v, and by $k_2(v)$ the position of the pair (ξ, u'') in v, where $k_2(v) > k_1(v)$. As γ is a complete set, $\gamma^\times$ must contain, for every member $u \in A'(\xi, u'')$, a string of pairs $v(u) = v_1 v_2 \cdots v_{k_1(v)} \cdots v_{k_2(v)-1}(\xi, u) v_{k_2(v)+1}(u) \cdots$ that starts with the same $k_2(v) - 1$ terms and has the pair (ξ, u) in position $k_2(v)$. Excising the segment $v_{k_1(v)} \cdots v_{k_2(v)-1}$, replace each such string of pairs by the shortened string $v'(u) := v_1 v_2 \cdots v_{k_1(v)-1}(\xi, u)\, v_{k_2(v)+1}(u) \cdots$ Repeat this process for every $u \in A'(\xi, u'')$. Denote by $\gamma_1^\times$ the resulting set of strings of state/input pairs, and let γ_1 be the corresponding set of input strings. As in Case 2, the length of some strings was reduced by this process, while no strings were made longer. Also, since all remaining branches of indeterminate transitions are not affected, it follows that γ_1 is still a complete set.

Case 4. $(\xi, u'), (\xi, u'') \in X \times A'$, i.e., u' and u'' are both pseudo input characters:
Then, (ξ, u') originates from an indeterminate pair of Σ, say from the indeterminate pair (z^i, u^i). Let $A'(\xi, u') = \{u^{i,1}, \ldots, u^{i,n(i)}\}$ be the set of all pseudo characters associated with the indeterminate pair (z^i, u^i). Similarly, (ξ, u'') originates from an indeterminate pair of Σ, say the pair (z^j, u^j); let $A'(\xi, u'') = \{u^{j,1}, \ldots, u^{j,n(j)}\}$ be the set of all pseudo characters associated with the indeterminate pair (z^j, u^j).
Consider now a member $v = v_1 v_2 \ldots v_{|v|} \in \gamma^\times$; denote by $k_1(v)$ the position of the pair (ξ, u') in v, and by $k_2(v)$ the position of the pair (ξ, u'') in v, where $k_2(v) > k_1(v)$. As γ is a complete set, $\gamma^\times$ must contain a response for every outcome of every indeterminate pair encountered; namely, for every member $u^2 \in A'(\xi, u'')$, there must be a string of pairs

$$v(u^2) = v_1 v_2 \cdots v_{k_1(v)-1}(\xi, u') v_{k_1(v)+1} \cdots v_{k_2(v)-1}(\xi, u^2) v_{k_2(v)+1}(u^2) \cdots$$

that starts with the same $k_2(v) - 1$ terms and has the pair (ξ, u^2) in position $k_2(v)$. Excising the segment $(\xi, u') v_{k_1(v)+1} \cdots v_{k_2(v)-1}$, we obtain the set of shortened strings

$$\gamma^- := \{ v_1 v_2 \cdots v_{k_1(v)-1}(\xi, u^2) v_{k_2(v)+1}(u^2) \,|\, u^2 \in A'(\xi, u'') \}.$$

Next, using again the fact that γ is a complete set, it follows that $\gamma^\times$ must include, for every member $u^1 \in A'(\xi, u')$, a string with the prefix $v(u^1) := v_1 v_2 \cdots v_{k_1(v)-1}(\xi, u^1)$. Denote by δ the set of all members of $\gamma^\times$ that have this prefix, namely,

$$\delta := \{ v \in \gamma^\times \,|\, v \text{ has a prefix } v(u^1) \text{ for some } u^1 \in A'(\xi, u') \}.$$

Now, delete all members of δ from $\gamma^\times$ and, thereafter, add the members of γ^- to the resulting new set; denote by $\gamma_1^\times$ the set of strings of state/input pairs obtained by this process. Let γ_1 be the set of input strings corresponding to $\gamma_1^\times$. As in Case 2, this process shortens some of the input strings of γ, while not making any strings longer. Also, since all branches of indeterminate transitions that remain in the set are preserved in the form they had in γ, it follows that γ_1 is still a complete set.

Finally, if still $|\gamma_1| \geq n$, perform on γ_1 the shortening operations performed on γ above. This process can be continued until a complete set of length not exceeding $n - 1$ is obtained. □

3.2. Reduced Sets of Input Sequences

In view of Theorem 3.2, a complete set of strings indicates the existence of a state feedback controller that generates a determinate transition between two states of an indeterminate asynchronous machine; the transition may or may not include indeterminate segments. Now, consider two states x and x' of the adjusted machine Σ_a, and let $S \subseteq (\tilde{A})^*$ be the set of all strings that take Σ_a from a stable combination with x to a stable combination with x'. If S includes a complete set γ, then there is a state feedback controller that implements a determinate transition from x to x', irrespective of the outcomes of indeterminate transitions that may be encountered along the way. Clearly, the presence or absence of additional members of S outside of γ has no implications in this regard. Thus, when concentrating on the existence of state feedback controllers that achieve determinate transitions, it is convenient to ignore members of S outside of γ. Doing so simplifies calculations and reduces clutter. This motivates the following definition of addition with a complete set in the semiring $\mathfrak{A}$:

$$\gamma + \alpha = \gamma \text{ for any complete set } \gamma \text{ and any set of strings } \alpha \subseteq (\tilde{A})^*. \tag{3.6}$$

A brief examination shows that the concatenation of two complete sets yields another complete set, and the following is valid.

Proposition 3.4: *Let Σ_a be an adjusted machine, and let x, x', and x'' be states of Σ_a. Let γ and γ' be complete sets of sequences, where γ takes Σ_a from a stable combination with x to a stable combination with x', while γ' takes Σ_a from a stable combination with x' to a stable combination with x''. Then, the concatenation $\gamma\gamma'$ is a complete set that takes Σ_a from a stable combination with x to a stable combination with x''.*

Proposition 3.4 and Equation (3.6) help us simplify expressions and remove superfluous terms without compromising information about the existence of state feedback controllers that induce determinate transitions. Note that (3.6) also applies when γ is a single string of characters of the original alphabet A, namely, when $\gamma \in \Gamma^0(\Sigma)$.

Example 3.5: Consider the adjusted machine Σ_a of Example 2.4. According to Table 2, the set of input strings S that take Σ_a from the state x^2 to the state x^1 is

$$S = \{c^1 + c^2b + da^2b + \cdots\}.$$

Now, according to (3.4), the combination $c^1 + c^2b$ is a complete set of strings. Thus, using (3.6), we can write $S = \{c^1 + c^2b\}$, a significant simplification which, by Theorem 3.2, indicates that a determinate transition from x^2 to x^1 can be implemented by a state feedback controller. □

Definition 3.6: A set of input strings $S \subseteq (\tilde{A})^+$ is *reducible* if it can be simplified into a complete set by using (3.6) and Proposition 3.4; otherwise, S is *irreducible*. A reducible set is in *reduced form* when it is expressed as a complete set. □

In view of our earlier discussion and Theorem 3.2, reducible sets represent transitions that can be implemented in determinate form by a state feedback controller. Therefore, the problem of determining the existence of such a state feedback controller can be resolved through simple algebraic manipulations within the ring $\mathfrak{A}$. This proves an important point of our discussion, and we state it in the following.

Theorem 3.7: *Let Σ be an asynchronous machine with adjusted machine Σ_a, and let $S(x, x') \subseteq (\tilde{A})^+$ be the set of all strings that take Σ_a from a stable combination with a state x to a stable combination with a state x'. Then, the following two statements are equivalent.*

(i) There is a state feedback controller that takes Σ through a determinate transition from x to x' in fundamental mode operation.
(ii) $S(x, x')$ is a reducible set. □

Note that the reduced form of a set is, in general, not unique.

4. Stable Reachability

4.1. Determinate and Indeterminate Transitions

The matrix of stable transitions of [13] characterizes the stable transitions of an asynchronous machine; it plays a critical role in the process of solving a variety of control problems. In the present section, we extend the definition of this matrix to machines that include indeterminate transitions. With this in mind, consider an indeterminate asynchronous machine $\Sigma = (A, X, x^0, f)$ with a state set $X = \{x^1, \ldots, x^n\}$ of n states, and let $\Sigma_a = (\tilde{A}, X, x^0, s_a)$ be the adjusted machine associated with Σ. Denote by $(\tilde{A})^{(n-1)}$ the set of all strings consisting of $n-1$ or fewer characters of the extended alphabet $\tilde{A}$. For two integers $p, q \in \{1, \ldots, n\}$ and an integer $i > 0$, define the set of strings

$$\alpha^i(p, q) := \left\{u \in (\tilde{A})^{(i)} \,\middle|\, s_a(x^p, u) = x^q\right\},$$

and let N be a character not in the alphabet $\tilde{A}$.

Definition 4.1: Let Σ be an indeterminate asynchronous machine with n states. The *adjusted matrix of stable transitions* $R^a(\Sigma)$ is the $n \times n$ matrix whose (p, q) entry is

$$R^a_{pq}(\Sigma) := \begin{cases} \alpha^{n-1}(p, q) & \text{if } \alpha^{n-1}(p, q) \neq \varnothing, \\ N & \text{else,} \end{cases}$$

$p, q = 1, 2, \ldots, n.$

The *one-step adjusted matrix of stable transitions* $R^a(\Sigma, 1)$ is the $n \times n$ matrix with the entries

$$R^a_{pq}(\Sigma, 1) := \begin{cases} \alpha^1(p, q) & \text{if } \alpha^1(p, q) \neq \varnothing, \\ N & \text{else,} \end{cases}$$

$p, q = 1, 2, \ldots, n.$ □

We extend the semiring $\mathfrak{A}$ to handle the character N by setting

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

Then, with the operations of the semiring $\mathfrak{A}$, we can use the usual definition of matrix multiplication to obtain the powers $(R^a(\Sigma, 1))^i, i = 1, 2, \ldots$, and we can construct the combination

$$(R^a(\Sigma))^{(i)} := R^a(\Sigma, 1) + (R^a(\Sigma, 1))^2 + \cdots + (R^a(\Sigma, 1))^i.$$

It can be seen that

$$R^a(\Sigma) = (R^a(\Sigma, 1))^{(n-1)} \tag{4.1}$$

(compare to [13]).

Example 4.2: For the machine of Example 2.4, we have

$$R^a(\Sigma, 1) = \begin{pmatrix} \{b+c+c^1\} & \{a\} & N & N \\ \{c^1\} & \{a+a^1\} & \{d\} & \{c^2\} \\ N & \{a^1\} & \{d\} & \{a^2\} \\ \{b\} & N & N & \{a+c+d+a^2+c^2\} \end{pmatrix}.$$

□

In the case of a determinate machine Σ, there are no pseudo input characters, so $\tilde{A} = A$, and Definition 4.1 reduces to the definition of the matrix of stable transitions given in [13]. The latter publication proves a statement analogous to the following.

Proposition 4.3: *Let Σ_a be an adjusted asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$, the extended input alphabet $\tilde{A}$, and the adjusted matrix of stable transitions $R^a(\Sigma)$. Then, the following two statements are equivalent.*

(i) There is an input string $u \in (\tilde{A})^+$ that takes Σ_a from a stable combination with the state x^i to a stable combination with the state x^j in fundamental mode operation.

(ii) $R^a_{ij}(\Sigma) \neq N$. □

Example 4.4: Using (4.1), we calculate the adjusted matrix of stable transitions for the machine Σ of Example 2.4 and obtain the following.

$$R^a(\Sigma) = \begin{pmatrix} R^a_{11}(\Sigma) & R^a_{12}(\Sigma) & R^a_{13}(\Sigma) & R^a_{14}(\Sigma) \\ R^a_{21}(\Sigma) & R^a_{22}(\Sigma) & R^a_{23}(\Sigma) & R^a_{24}(\Sigma) \\ R^a_{31}(\Sigma) & R^a_{32}(\Sigma) & R^a_{33}(\Sigma) & R^a_{34}(\Sigma) \\ R^a_{41}(\Sigma) & R^a_{42}(\Sigma) & R^a_{43}(\Sigma) & R^a_{44}(\Sigma) \end{pmatrix},$$

where

$$\begin{aligned} R^a_{11}(\Sigma) = \{&aa^1c^1 + ac^1 + ac^1b + ac^1c + ac^2b + b \\ &+ bac^1 + bc + bcb + bcc^1 + bc^1 + bc^1b \\ &+ bc^1c + c + cac^1 + cb + cbc + cbc^1 + cc^1 \\ &+ cc^1b + cc^1c + c^1 + c^1ac^1 + c^1b + c^1bc \\ &+ c^1bc^1 + c^1c + c^1cb + c^1cc^1\} \end{aligned}$$

$$\begin{aligned} R^a_{12}(\Sigma) = \{&a + aa^1 + aa^1a + ac^1a + ada^1 + ba + baa^1 \\ &+ bca + bc^1a + ca + caa^1 + cba + c^1a \\ &+ c^1aa^1 + c^1ba + c^1ca\} \end{aligned}$$

$$R^a_{13}(\Sigma) = \{aa^1d + ad + bad + cad + c^1ad\}$$

$$\begin{aligned} R^a_{14}(\Sigma) = \{&aa^1c^2 + ac^2 + ac^2a + ac^2a^2 + ac^2c + ac^2d \\ &+ ada^2 + bac^2 + cac^2 + c^1ac^2\} \end{aligned}$$

$$\begin{aligned} R^a_{21}(\Sigma) = \{&aa^1c^1 + ac^1 + ac^1b + ac^1c + ac^2b + a^1ac^1 \\ &+ a^1c^1 + a^1c^1b + a^1c^1c + a^1c^2b + c^1 \\ &+ c^1ac^1 + c^1b + c^1bc + c^1bc^1 + c^1c + c^1cb \\ &+ c^1cc^1 + c^2ab + c^2a^2b + c^2b + c^2bc \\ &+ c^2bc^1 + c^2cb + c^2db + da^1c^1 + da^2b\} \end{aligned}$$

$$\begin{aligned} R^a_{22}(\Sigma) = \{&a + aa^1 + aa^1a + ac^1a + ada^1 + a^1 + a^1a \\ &+ a^1aa^1 + a^1c^1a + a^1da^1 + c^1a + c^1aa^1 \\ &+ c^1ba + c^1ca + c^2ba + da^1 + da^1a\} \end{aligned}$$

$$R^a_{23}(\Sigma) = \{aa^1d + ad + a^1ad + a^1d + c^1ad + d + da^1d\}$$

$$\begin{aligned} R^a_{24}(\Sigma) = \{&aa^1c^2 + ac^2 + ac^2a + ac^2a^2 + ac^2c + ac^2d \\ &+ ada^2 + a^1ac^2 + a^1c^2 + a^1c^2a + a^1c^2a^2 \\ &+ a^1c^2c + a^1c^2d + a^1da^2 + c^1ac^2 + c^2 \\ &+ c^2a + c^2aa^2 + c^2ac + c^2ac^2 + c^2ad \\ &+ c^2a^2 + c^2a^2a + c^2a^2c + c^2a^2c^2 + c^2a^2d \\ &+ c^2c + c^2ca + c^2ca^2 + c^2cc^2 + c^2cd + c^2d \\ &+ c^2da + c^2da^2 + c^2dc + c^2dc^2 + da^1c^2 \\ &+ da^2 + da^2a + da^2c + da^2c^2 + da^2d\} \end{aligned}$$

$$\begin{aligned} R^a_{31}(\Sigma) = \{&a^1ac^1 + a^1c^1 + a^1c^1b + a^1c^1c + a^1c^2b \\ &+ a^2ab + a^2b + a^2bc + a^2bc^1 + a^2cb \\ &+ a^2c^2b + a^2db + da^1c^1 + da^2b\} \end{aligned}$$

$$\begin{aligned} R^a_{32}(\Sigma) = \{&a^1 + a^1a + a^1aa^1 + a^1c^1a + a^2ba \\ &+ da^1 + da^1a\} \end{aligned}$$

$$
\begin{aligned}
R^a_{33}(\Sigma) = \{&a^1ad + a^1d + d + da^1d\}\\
R^a_{34}(\Sigma) = \{&a^1ac^2 + a^1c^2 + a^1c^2a + a^1c^2a^2 + a^1c^2c\\
&+ a^1c^2d + a^2 + a^2a + a^2aa^2 + a^2ac\\
&+ a^2ac^2 + a^2ad + a^2c + a^2ca + a^2ca^2\\
&+ a^2cc^2 + a^2cd + a^2c^2 + a^2c^2a + a^2c^2a^2\\
&+ a^2c^2c + a^2c^2d + a^2d + a^2da + a^2da^2\\
&+ a^2dc + a^2dc^2 + da^1c^2 + da^2 + da^2a\\
&+ da^2c + da^2c^2 + da^2d\}\\
R^a_{41}(\Sigma) = \{&aa^2b + ab + abc + abc^1 + acb + ac^2b\\
&+ adb + a^2ab + a^2b + a^2bc + a^2bc^1 + a^2cb\\
&+ a^2c^2b + a^2db + b + bac^1 + bc + bcb\\
&+ bc^1 + bc^1b + bc^1c + bcc^1 + cab + ca^2b\\
&+ cb + cbc + cbc^1 + cc^2b + cdb + c^2ab\\
&+ c^2a^2b + c^2b + c^2bc + c^2bc^1 + c^2cb\\
&+ c^2db + dab + da^2b + db + dbc\\
&+ dbc^1 + dcb + dc^2b\}\\
R^a_{42}(\Sigma) = \{&aba + a^2ba + ba + baa^1 + bca + bc^1a\\
&+ cba + c^2ba + dba\}\\
R^a_{43}(\Sigma) = \{&bad\}\\
R^a_{44}(\Sigma) = \{&a + aa^2 + aa^2a + aa^2c + aa^2c^2 + aa^2d\\
&+ ac + aca + aca^2 + acc^2 + acd + ac^2\\
&+ ac^2a + ac^2a^2 + ac^2c + ac^2d + ad\\
&+ ada + ada^2 + adc + adc^2 + a^2 + a^2a\\
&+ a^2aa^2 + a^2ac + a^2ac^2 + a^2ad + a^2c\\
&+ a^2ca + a^2ca^2 + a^2cc^2 + a^2cd + a^2c^2\\
&+ a^2c^2a + a^2c^2a^2 + a^2c^2c + a^2c^2d + a^2d\\
&+ a^2da + a^2da^2 + a^2dc + a^2dc^2 + bac^2\\
&+ c + ca + caa^2 + cac + cac^2 + cad + ca^2\\
&+ ca^2a + ca^2c + ca^2c^2 + ca^2d + cc^2 + cc^2a\\
&+ cc^2a^2 + cc^2c + cd + cda + cda^2 + cdc\\
&+ cdc^2 + c^2 + c^2a + c^2aa^2 + c^2ac + c^2ac^2\\
&+ c^2ad + c^2a^2 + c^2a^2a + c^2a^2c + c^2a^2c^2\\
&+ c^2a^2d + c^2c + c^2ca + c^2ca^2 + c^2cc^2\\
&+ c^2cd + cc^2d + c^2d + c^2da + c^2da^2\\
&+ c^2dc + c^2dc^2 + d + da + daa^2\\
&+ dac + dac^2 + dad + da^2 + da^2a\\
&+ da^2c + da^2c^2 + da^2d + dc + dca\\
&+ dca^2 + dcc^2 + dcd + dc^2 + dc^2a + dc^2a^2\\
&+ dc^2c + dc^2d + dc^2\}. \square
\end{aligned}
$$

The entries of the adjusted matrix of stable transitions can be simplified by reduction, and this leads us to the following notion.

Definition 4.5: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with the adjusted matrix of stable transitions $R^a(\Sigma)$. The *reduced matrix of stable transitions* $R(\Sigma)$ is obtained when every reducible entry of $R^a(\Sigma)$ is expressed in reduced form; entries that are not reducible are left in their original form. □

As the reduced form of a set is not unique, neither is the reduced matrix of stable transitions. In view of Propositions 4.3 and 3.3, the reduced matrix of stable transitions characterizes all determinate stable transitions that can be achieved in the machine Σ by state feedback controllers that operate in fundamental mode, as follows.

Corollary 4.6: *Let Σ be an asynchronous machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the reduced matrix of stable transitions $R(\Sigma)$. Then, the following two statements are equivalent for every pair of states $x^i, x^j \in X$.*

(i) There is a state feedback controller that takes Σ through a determinate transition from a stable combination with x^i to a stable combination with x^j in fundamental mode operation.
(ii) $R_{ij}(\Sigma)$ is a complete set. □

Example 4.7: Reducing the adjusted matrix of stable transitions of Example 4.4, we obtain a reduced matrix of stable transitions of the machine Σ of Example 2.4:

$$
R(\Sigma) = \begin{pmatrix} \{b\} & \{a\} & \{ad\} & R_{14}(\Sigma) \\ \{\gamma_1\} & \{a\} & \{d\} & R_{24}(\Sigma) \\ \{\gamma_2\} & \{\gamma_3\} & \{d\} & R_{34}(\Sigma) \\ \{b\} & \{ba\} & \{bad\} & \{a\} \end{pmatrix},
$$

where $\gamma_1 := c^1 + c^2b$, $\gamma_2 = a^1\gamma_1 + a^2b$, and $\gamma_3 = a^1 + a^2ba$ are complete sets; the entries $R_{14}(\Sigma), R_{24}(\Sigma)$, and $R_{34}(\Sigma)$ are irreducible and are given in Example 4.4. □

The information contained in the reduced matrix of stable transitions can be further condensed by the following notion (compare to [13]).

Definition 4.8: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with n states and with the reduced matrix of stable transitions $R(\Sigma)$. Let Δ be a character not in the set $\tilde{A} \cup \{N\}$. Then, the *skeleton matrix* $K(\Sigma)$ is an $n \times n$ matrix with the entries

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

$i, j = 1, 2, \ldots, n$. □

In the skeleton matrix, transitions indicated by 1 can be implemented in determinate form by a state feedback controller operating in fundamental mode; transitions indicated by 0 are impossible; and transitions indicated by Δ are indeterminate – they may or may not be possible, depending on the outcomes of indeterminate transitions along the way.

Example 4.9: Based on the reduced matrix of stable transitions of Example 4.7, the skeleton matrix of the machine Σ of Example 2.4 is given by

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

4.2. Connected Sets of States

An asynchronous machine may, and often does, have a large number of states. The following notion helps reduce the computational burden of calculating controllers for a machine with many states.

Definition 4.10: Let $\Sigma = (A, X, x^0, f)$ be an input/state asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and the skeleton matrix $K(\Sigma)$. A subset χ of states is a *connected set* if $K_{ij}(\Sigma) = K_{ji}(\Sigma) = 1$ for every pair of states $x^i, x^j \in \chi$. The connected set that includes the initial state x^0 is the *initial connected set* χ^0. □

A connected set of the machine Σ is characterized by the fact that every two of its members are mutually stably reachable from each other through a determinate transition, possibly driven by a state feedback controller operating in fundamental mode. Note that membership in a connected set is an equivalence relation. Indeed, the relation is reflexive, since every state has a stable combination and hence belongs to its own connected set; the relation is commutative, since, by definition, x^i is connected to x^j if and only if x^j is connected to x^i; and, finally, by Proposition 3.4, the relation is transitive

$$K(\Sigma) = \begin{pmatrix} 1 & 1 & 1 & \cancel{\Delta} \\ 1 & 1 & 1 & \cancel{\Delta} \\ 1 & 1 & 1 & \cancel{\Delta} \\ \cancel{1} & \cancel{1} & \cancel{1} & \cancel{1} \end{pmatrix}$$

Fig. 4. Finding a connected set.

as well. Thus, the state set $X = \{x^1, x^2, \ldots, x^n\}$ of Σ is partitioned into a union of disjoint connected sets, say $X = \chi^1 \cup \chi^2 \cup \ldots \cup \chi^m$, where $m \leq n$. Of course, some of these connected sets may include just a single state, while others may include many states. The following statement describes a simple way to find the largest connected set to which a particular state belongs.

Proposition 4.11: *Let $\Sigma = (A, X, x^0, f)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and the skeleton matrix $K(\Sigma)$. Fix an integer $i \in \{1, 2, \ldots, n\}$. For every $j \in \{1, 2, \ldots, n\}$ for which $K_{ij}(\Sigma) \neq 1$ or $K_{ji}(\Sigma) \neq 1$, cross out column j and row j of $K(\Sigma)$. Then, the remaining columns (or, equivalently, the remaining rows) of $K(\Sigma)$ represent the largest connected set that includes x^i.*

Proof: Let χ be the largest connected set that includes the state x^i. By Definition 4.10, a state x^j is a member of χ if and only if $K_{ij}(\Sigma) = 1$ and $K_{ji}(\Sigma) = 1$. The proposition follows directly from this fact. □

Example 4.12: Applying the process of Proposition 4.11 to the skeleton matrix of Example 4.9, we obtain the outcome for the state x^1 as in Fig. 4. From Fig. 4, it follows that the largest connected set that includes x^1 is $\chi = \{x^1, x^2, x^3\}$. □

By definition, members of a connected set are all stably reachable from each other through determinate transitions (possibly driven by a state feedback controller operating in fundamental mode). As a result, when one member of a connected set χ is stably reachable from a state x of Σ by a determinate transition, then every member of χ is stably reachable from x by a determinate transition; and vice versa: when a state x is stably reachable from one member of χ by a determinate transition, then x is stably reachable from every member of χ by a determinate transition. Accordingly, when discussing stable reachability, each connected set can be considered as one integral unit. This leads us to the following notion.

Table 3. The recursion function of the diminished realization.

Z	a	b	c	d	a^1	a^2	c^1	c^2
z^1	z^1	z^1	z^1	z^1	z^1	z^2	z^1	z^2
z^2	z^2	z^1	z^2	z^2	–	z^2	–	z^2

Definition 4.13: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with the adjusted machine $\Sigma_a = (\tilde{A}, X, x^0, s_a)$ and the class of connected sets $\{\chi^1, \ldots, \chi^m\}$. Let $Z = \{z^1, z^2, \ldots, z^m\}$ be a set of m elements, and let $P(Z)$ be the family of subsets of Z. A *diminished realization* $(\tilde{A}, Z, z^0, \sigma)$ of Σ has the input alphabet $\tilde{A}$; the state set Z, where z^i represents the connected set χ^i, $i = 1, 2, \ldots, m$; and the initial condition $z^0 \in Z$ that represents the initial connected set. For an element $z \in Z$, denote by $\chi(z)$ the connected set of states corresponding to z.

A pair $(z, u) \in Z \times \tilde{A}$ is *valid* if (x, u) is a valid pair of Σ_a for a state $x \in \chi(z)$. The recursion function $\sigma : Z \times \tilde{A} \to P(Z)$ is a partial function defined for all valid pairs $(z, u) \in Z \times \tilde{A}$ by

$$\sigma(z, u) := \left\{ z' \in Z \mid s_a(x, u) \in \chi(z') \text{ for a state } x \in \chi(z) \right\}.$$

□

The recursion function σ of Definition 4.13 is usually set valued: its values consist of all connected sets that can be reached from its argument. A diminished realization may have a smaller number of states than the original realization. As a result, computations with diminished realizations may have a lower computational complexity. In the sequel, we use diminished realizations to analyze control features of indeterminate asynchronous machines.

Example 4.14: For the machine Σ of Example 2.4, we have seen in Example 4.12 that the family of connected sets consists of the two members

$$\chi^1 := \{x^1, x^2, x^3\} \text{ and } \chi^2 := \{x^4\}.$$

Using the diminished state set $Z = \{z^1, z^2\}$, we obtain the recursion function σ of the diminished realization as shown in Table 3. □

The *diminished matrix of stable transitions* $R^d(\Sigma)$, the *one-step diminished matrix of stable transitions* $R^d(\Sigma, 1)$, and the *diminished skeleton matrix* $K^d(\Sigma)$ of an asynchronous machine Σ are defined, respectively, as the reduced matrix of stable transitions, the one-step matrix of stable transitions, and the skeleton matrix of a diminished realization of Σ. All three matrices are $m \times m$ rather than $n \times n$, where m is the number of connected sets, while $n \geq m$ is the number of states of Σ. These matrices can be derived either by combining blocks corresponding to connected sets in the reduced matrices of the original machine Σ, or by direct use of the diminished stable recursion function of Σ. The former method is described in the following statement.

Proposition 4.15: *Let Σ be an indeterminate asynchronous machine with state set $X = \{x^1, \ldots, x^n\}$, diminished state set $\{z^1, \ldots, z^m\}$, matrix of stable transitions $R(\Sigma)$, and skeleton matrix $K(\Sigma)$. Let $\chi(z^i)$ be the connected set of states corresponding to the diminished state z^i. Then, for each pair of integers $i, j \in \{1, 2, \ldots, m\}$, the corresponding entries of the diminished matrices are given by*

(i) $R^d_{ij}(\Sigma) = \bigcup_{\substack{\{p : x^p \in \chi(z^i)\} \\ \{q : x^q \in \chi(z^j)\}}}$ $R_{pq}(\Sigma)$ *written in reduced form*; and

(ii) $$K^d_{ij}(\Sigma) = \begin{cases} 1 & \text{if } K_{pq}(\Sigma) = 1 \text{ for any pair of states } x^p \in \chi(z^i), x^q \in \chi(z^j), \\ \Delta & \text{if } K_{pq}(\Sigma) = \Delta \text{ for any pair of states } x^p \in \chi(z^i), x^q \in \chi(z^j), \\ 0 & \text{if } K_{pq}(\Sigma) = 0 \text{ for any pair of states } x^p \in \chi(z^i), x^q \in \chi(z^j). \end{cases}$$

Proof: Let s_a be the adjusted stable recursion function of Σ and let σ be the diminished stable recursion function. For two integers $i, j \in \{1, 2, \ldots, m\}$, consider two states $x^p \in \chi(z^i)$ and $x^q \in \chi(z^j)$ of Σ. By Definition 4.13, a string $u \in (\tilde{A})^+$ satisfies $s_a(x^p, u) = x^q$ if and only if $z^j \in \sigma(z^i, u)$. This implies *(i)*. Regarding *(ii)*, it follows from the definition of a connected set that the following are true: $K_{rt}(\Sigma) = K_{r't'}(\Sigma) = K^d_{ij}(\Sigma)$ for all pairs $(r, t), (r', t')$ for which $x^r, x^{r'} \in \chi(z^i)$ and $x^t, x^{t'} \in \chi(z^j)$. Hence, *(ii)* is valid and our proof is complete. □

Example 4.16: Referring to the machine Σ of Example 2.4, recall that, according to Example 4.14, the connected sets of Σ are $\chi(z^1) = \{x^1, x^2, x^3\}$ and $\chi(z^2) = \{x^4\}$. Then, applying Proposition 4.15*(i)* to the matrix $R(\Sigma)$ of Example 4.7 and reducing the resulting entries, we obtain

$$R^d(\Sigma) = \begin{pmatrix} R^d_{11}(\Sigma) & R^d_{12}(\Sigma) \\ R^d_{21}(\Sigma) & R^d_{22}(\Sigma) \end{pmatrix}, \tag{4.2}$$

where

$$R^d_{11}(\Sigma) = \bigcup_{\substack{p \in \{1,2,3\} \\ q \in \{1,2,3\}}} R_{pq}(\Sigma) = \{b\} \text{ (in reduced form)},$$

$$\begin{aligned}
R^d_{12}(\Sigma) = {} & \bigcup_{\substack{p\in\{1,2,3\}\\ q=4}} R_{pq}(\Sigma)\\
= {} & \{bac^2 + cac^2 + aa^1c^2 + ac^2 + ac^2a + ac^2a^2\\
& + ac^2c + ac^2d + ada^2 + a^1ac^2 + a^1c^2\\
& + a^1c^2a + a^1c^2a^2 + a^1c^2c + a^1c^2d + a^1da^2\\
& + c^1ac^2 + c^2 + c^2a + c^2a + c^2aa^2 + c^2ac\\
& + c^2ac^2 + c^2ad + c^2a^2 + c^2a^2a + c^2a^2c\\
& + c^2a^2c^2 + c^2a^2d + c^2c + c^2ca + c^2ca^2\\
& + c^2cc^2 + c^2cd + c^2d + c^2da + c^2da^2\\
& + c^2dc + c^2dc^2 + da^1c^2 + da^2 + da^2a\\
& + da^2c + da^2c^2 + da^2d + a^2 + a^2a + a^2aa^2\\
& + a^2ac + a^2ac^2 + a^2ad + a^2c + a^2ca\\
& + a^2ca^2 + a^2cc^2 + a^2cd + a^2c^2 + a^2c^2a\\
& + a^2c^2a^2 + a^2c^2c + a^2c^2d + a^2d + a^2da\\
& + a^2da^2 + a^2dc + a^2dc^2\}\ \text{(irreducible)},\\
R^d_{21}(\Sigma) = {} & \bigcup_{\substack{p=4\\ q\in\{1,2,3\}}} R_{pq}(\Sigma) = \{b\}\ \text{(in reduced form)},
\end{aligned}$$

and

$$R^d_{22}(\Sigma) = \bigcup_{\substack{p=4\\ q=4}} R_{pq}(\Sigma) = \{a\}\ \text{(in reduced form)}.$$

Similarly, applying Proposition 4.15*(ii)* to the matrix $K(\Sigma)$ of Example 4.9, we obtain

$$K^d(\Sigma) = \begin{pmatrix} 1 & \Delta \\ 1 & 1 \end{pmatrix}.$$

The same result can also be obtained, of course, from the matrix $R^d(\Sigma)$ listed above. □

The significance of the diminished skeleton matrix is brought to light in the next section.

5. Model Matching

We turn now to the construction of state feedback controllers that elicit desirable behavior from a given indeterminate asynchronous machine by guiding the closed loop system to match a prescribed model. To this end, let $\Sigma = (A, X, x^0, f)$ be an indeterminate input/state machine connected to a state feedback controller C as described in Fig. 1, and let $\Sigma' = (A, X, x^0, s')$ be a specified determinate model. The objective is to find a controller C for which the closed loop machine Σ_c exhibits stable-state behavior that simulates Σ'. For such a controller C, we write $\Sigma_c = \Sigma'$.

The model Σ' is a determinate stable state asynchronous machine with the same input alphabet and the same state set as the controlled machine Σ. Let $K(\Sigma)$ be the skeleton matrix of the machine Σ, and let $K(\Sigma')$ be the skeleton matrix of the model Σ'. When Σ has no indeterminate transitions, [13, Theorem 5.1] states that a controller C satisfying $\Sigma_c = \Sigma'$ exists if and only if

$$K(\Sigma) \geq K(\Sigma'), \tag{5.1}$$

where the inequality is taken entry-by-corresponding-entry.

When the machine Σ has indeterminate transitions, its skeleton matrix $K(\Sigma)$ may include entries of the character Δ; such entries indicate transitions whose outcome is indeterminate and not known a-priori. Clearly, when an entry of Δ in $K(\Sigma)$ appears in a position corresponding to an entry of 1 in $K(\Sigma')$, model matching cannot be guaranteed in advance. On the other hand, an entry of Δ in $K(\Sigma)$ that appears in a position corresponding to an entry of 0 in $K(\Sigma')$ has no direct bearing on model matching, since the transition it represents is not required when matching the model Σ'. All these considerations can be incorporated into the inequality (5.1) by considering Δ as a number satisfying

$$0 < \Delta < 1. \tag{5.2}$$

With the assignment (5.2), inequality (5.1) continues to serve as a necessary and sufficient condition for model matching when indeterminate transitions are present.

Once an indeterminate transition of Σ is tested and its outcome is recorded, some of the Δ entries of the skeleton matrix $K(\Sigma)$ may change: an entry of Δ turns into 0 if the outcome of the test shows that the corresponding transition is impossible; an entry of Δ turns into 1 if the outcome of the test shows that the corresponding transition is possible; and, finally, the entry remains Δ if the corresponding transition depends on other indeterminate transitions that have not yet been tested. Accordingly, for an indeterminate machine Σ, the skeleton matrix $K(\Sigma)$ may change with time, as outcomes of more and more indeterminate transitions become known.

Next, we show that condition (5.1) continues to remain valid when skeleton matrices are replaced by their diminished counterparts. This results in a potential reduction of the problem's dimensionality. First, however, we have to clarify what is meant by the diminished skeleton matrix of a model.

Definition 5.1: Let Σ be an input/state asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and a family

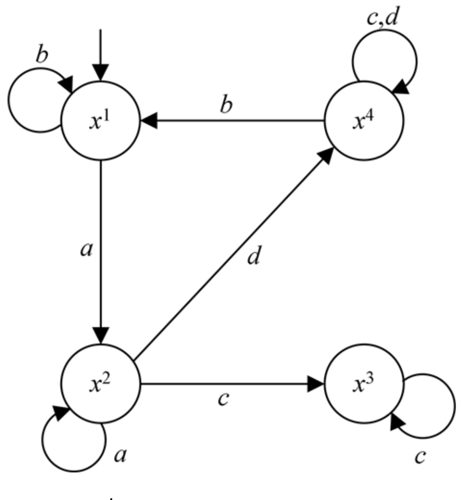

Fig. 5. The model Σ'.

of m connected sets $\{\chi^1, \ldots, \chi^m\}$. Let Σ' be a determinate input/state asynchronous machine with the same state set X, and let $K(\Sigma')$ be its skeleton matrix. The *diminished skeleton matrix* $K^d(\Sigma', \Sigma)$ *of* Σ' *relative to* Σ is an $m \times m$ matrix with the entries

$$K^d_{ij}(\Sigma', \Sigma) := \begin{cases} 1 & \text{if there are states } x^p \in \chi^i \text{ and} \\ & \quad x^q \in \chi^j \text{ for which } K_{pq}(\Sigma') = 1, \\ 0 & \text{otherwise,} \end{cases}$$

$i,j = 1, \ldots, m$. □

The diminished skeleton matrix of Σ' relative to Σ indicates the transitions that Σ' can make among connected sets of Σ. As we show shortly, these transitions among connected sets must also be possible for Σ, if Σ can be controlled to match the model Σ'.

Example 5.2: Consider the machine Σ of Example 2.4 with the model Σ' shown in Fig. 5.

A direct examination shows that the skeleton matrix of Σ' is

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

In Example 4.14, we have seen that the connected sets of the machine Σ are $\chi^1 = \{x^1, x^2, x^3\}$ and $\chi^2 = \{x^4\}$, so that $m = 2$ in this case. Applying Definition 5.1, we obtain

$$K^d(\Sigma', \Sigma) = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}. \square$$

Diminished skeleton matrices can be used to determine whether or not model matching is possible, as follows.

Proposition 5.3: *Let* Σ *be an asynchronous machine with the state set* $X = \{x^1, \ldots, x^n\}$ *and the family of connected sets* $\{\chi^1, \ldots, \chi^m\}$*, and let* Σ' *be a determinate asynchronous machine with the same state set X. Let* $K(\Sigma)$*,* $K(\Sigma')$*,* $K^d(\Sigma)$*, and* $K^d(\Sigma', \Sigma)$ *be the corresponding skeleton and diminished skeleton matrices. Then,* $K(\Sigma) \geq K(\Sigma')$ *if and only if* $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$.

Proof: Note that, by Definition 5.1, the matrix $K^d(\Sigma', \Sigma)$ has only entries of 0 or 1. Assume first that $K(\Sigma) \geq K(\Sigma')$, and consider a pair of integers $p, q \in \{1, 2, \ldots, n\}$ for which $K_{pq}(\Sigma') = 1$. Then, since $K(\Sigma) \geq K(\Sigma')$, we must have $K_{pq}(\Sigma) = 1$ as well. Now, let $i,j \in \{1, \ldots, m\}$ be integers such that $x^p \in \chi^i$ and $x^q \in \chi^j$. In view of Proposition 4.15 and Definition 5.1, we have $K^d_{ij}(\Sigma) = 1$ and $K^d_{ij}(\Sigma', \Sigma) = 1$, so that $K^d_{ij}(\Sigma) \geq K^d_{ij}(\Sigma', \Sigma)$. As the latter is true for all $i,j \in \{1, \ldots, m\}$, we conclude that $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$.

Conversely, assume that $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$ and consider a pair of integers $i,j \in \{1, \ldots, m\}$ for which $K^d_{ij}(\Sigma', \Sigma) = 1$. As $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$, it follows that $K^d_{ij}(\Sigma) = 1$ as well. By Proposition 4.15, the latter implies that there is a pair of states $x^{p'} \in \chi^i$ and $x^{q'} \in \chi^j$ for which $K_{p'q'}(\Sigma) = 1$; by Definition 4.10, this entails that $K_{pq}(\Sigma) = 1$ for all integers $p, q \in \{1, 2, \ldots, n\}$ for which $x^p \in \chi^i$ and $x^q \in \chi^j$. But then, we have $K_{pq}(\Sigma) \geq K_{pq}(\Sigma')$ for all $p, q \in \{1, 2, \ldots, n\}$ for which $x^p \in \chi^i$ and $x^q \in \chi^j$. Finally, since this conclusion is valid for all $i,j \in \{1, \ldots, m\}$, it follows that $K(\Sigma) \geq K(\Sigma')$, and our proof concludes. □

The use of diminished skeleton matrices allows us to analyze and solve control problems in lower dimensionality, and hence with lower computational effort. Combining Proposition 3.3 with the discussion following (5.1), we obtain the following.

Theorem 5.4: *Let* $\Sigma = (A, X, x^0, f)$ *be an asynchronous machine with the diminished skeleton matrix* $K^d(\Sigma)$*, let* $\Sigma' = (A, X, x^0, s')$ *be a determinate stable state machine serving as a model, and let* $K^d(\Sigma', \Sigma)$ *be the diminished skeleton matrix of* Σ' *relative to* Σ*. Then, the following two statements are equivalent.*

(i) There is a state feedback controller C for which $\Sigma_c = \Sigma'$*, where* Σ_c *operates in fundamental mode.*
(ii) $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$. □

The most convenient way to determine whether model matching is possible is by examining the inequality of

Theorem 5.4*(ii)*. This inequality may fail under the following circumstances.

1. $K^d(\Sigma)$ has an entry of 0 in a position in which $K^d(\Sigma', \Sigma)$ has an entry of 1; or
2. $K^d(\Sigma)$ has an entry of Δ in a position in which $K^d(\Sigma', \Sigma)$ has an entry of 1.

In Case 1, model matching is definitely impossible. However, in Case 2, the feasibility of model matching depends on the outcome of one or more indeterminate transitions of Σ; if the outcome turns out favorably, then model matching is possible.

To handle Case 2, the controller C of Fig. 1 can be designed to test automatically the outcomes of certain indeterminate transitions. As long as such testing is confined to transient transitions of the closed loop machine Σ_c, it does not interfere with user experience. This aspect of controller design is discussed in the next two sections.

6. Comparison and Reduction

When controlling indeterminate asynchronous machines, it is instrumental to develop controllers that automatically test indeterminate transitions and record their outcomes. Such controllers must satisfy two requirements: *(i)* they must operate in fundamental mode when combined with the controlled machine; and *(ii)* they must not disturb user experience. We start with an examination of the first requirement.

6.1. Detectability

Consider an indeterminate asynchronous machine $\Sigma = (A, X, x^0, f)$. Recall from Definition 2.1 that the recursion function f of Σ is a set valued function whose values indicate the possible outcomes of indeterminate transitions. Let $(x, u) \in X \times A$ be a valid indeterminate pair of Σ; then, $f(x, u)$ is its set of possible outcomes. A troublesome case occurs when the set $f(x, u)$ includes the state x itself together with another state, say, $z \neq x$. In such case, (x, u) is a stable combination if the outcome of the transition is x; however, (x, u) is a transient combination if the outcome of the transition is z. Consequently, when encountering the pair (x, u) for the first time, it is impossible to tell whether Σ is in a stable combination or not. This makes fundamental mode operation impossible. Thus, in order to preserve fundamental model operation, we cannot access any indeterminate pair (x, u) whose set of potential outcomes includes the state x itself. This leads us to the following.

Definition 6.1: Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine. A valid pair $(x, u) \in X \times A$ is *detectable* if either $x \notin f(x, u)$ or $f(x, u) = \{x\}$. □

In view of the previous paragraph, fundamental mode operation of an indeterminate asynchronous machine is impossible if a non-detectable pair is activated. To assure that the latter does not occur, we adjust the domain of the machine's recursion function by reclassifying all non-detectable pairs as invalid pairs; this effectively prohibits access of non-detectable pairs.

Convention 6.2: *For asynchronous machines considered in this paper, all non-detectable pairs are reclassified as invalid pairs.* □

Convention 6.2 guarantees that detectability is not an obstacle in the remaining parts of this paper and, on this account, we shall ignore detectability altogether from now on.

6.2. Comparison Matrices

When an indeterminate asynchronous machine $\Sigma = (A, X, x^0, f)$ is inserted into the configuration of Fig. 1, the controller C may automatically test and record the outcomes of some indeterminate transitions of Σ. Such a testing and recording process is intended to improve the performance of the closed loop machine Σ_c; it must be performed without interfering with user experience. We proceed to develop an algorithm that governs the controller C during such testing.

For the sake of simplicity, all testing considered in this paper is performed immediately upon activation of the closed loop machine Σ_c, before any commands are applied at the external input ω of Fig. 1. All testing starts and ends with Σ being in its initial state x^0. Similar testing can be repeated at every stable combination of Σ.

To avoid disturbing user experience, all testing must be confined to strings of transient transitions of the closed loop machine Σ_c. And, since we restrict ourselves to testing in the initial state x^0 of Σ, all testing strings must start with Σ being at x^0 and must return Σ with certainty to x^0 at the end of testing. This means that testing must be confined to complete sets of strings that take the adjusted machine Σ_a in round trips that start and end at x^0. Our first objective is to determine which indeterminate transitions of Σ must be tested.

Consider the problem of controlling the machine Σ to match a model Σ'. Let $K^d(\Sigma)$ be the $m \times m$ diminished skeleton matrix of Σ and let $K^d(\Sigma', \Sigma)$ be the diminished skeleton matrix of Σ' with respect to Σ. In view of Theorem 5.4, the existence of a solution of the model matching problem depends on a favorable comparison of these two skeleton matrices. Clearly, only entries of $K^d(\Sigma)$ that are juxtaposed to entries of 1 in $K^d(\Sigma', \Sigma)$ need to be examined. Of these, entries of 0 are prohibitive to model matching, while entries of 1 are permissive;

entries of Δ, on the other hand, impart an uncertainty that can be resolved only through testing of indeterminate transitions of Σ. To help single out the indeterminate transitions that must be tested, define the set of pairs

$$D'(\Sigma',\Sigma) := \Big\{(i,j) \in \{1,\dots,m\} \times \{1,\dots,m\} \Big| \, K^d_{ij}(\Sigma) = \Delta \text{ and } K^d_{ij}(\Sigma',\Sigma) = 1\Big\},$$

and set

$$D(\Sigma',\Sigma) := \begin{cases} \varnothing & \text{if } K^d_{ij}(\Sigma) = 0 \text{ while } K^d_{ij}(\Sigma',\Sigma) = 1 \\ & \quad \text{for some } i,j \in \{1,\dots,m\}; \\ D'(\Sigma',\Sigma) & \text{otherwise.} \end{cases} \tag{6.1}$$

Example 6.3: Consider the machine Σ of Example 2.4 with the model Σ' of Example 5.2. Using (6.1) with the matrix $K^d(\Sigma)$ of Example 4.16 and the matrix $K^d(\Sigma',\Sigma)$ of Example 5.2, we obtain $D(\Sigma',\Sigma) = \{(1,2)\}$. □

Now, recall the pseudo alphabet A' of (2.4), and denote by $P(S)$ the class of all subsets of a set S. Define the projection $\Pi' : (\tilde{A})^+ \to P(A')$ that extracts all pseudo characters from strings $t \in (\tilde{A})^+$, namely,

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

For a list of strings $t_1, t_2, \dots, t_q \in (\tilde{A})^+$, the projection Π' creates a list of sets, where each set consists of the pseudo characters included in one of the strings:

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

We use commas to separate the members of the resulting list. Duplicate members are listed only once.

Example 6.4: For the entry $R^d_{12}(\Sigma)$ of (4.2) we have

$$\begin{aligned}\Pi' R^d_{12}(\Sigma) = \Pi'\Big\{ & bac^2 + cac^2 + aa^1c^2 + ac^2 + ac^2a \\ & + ac^2a^2 + ac^2c + ac^2d + ada^2 + a^1ac^2 \\ & + a^1c^2 + a^1c^2a + a^1c^2a^2 + a^1c^2c \\ & + a^1c^2d + a^1da^2 + c^1ac^2 + c^2 \\ & + c^2a + c^2a + c^2aa^2 + \cdots\Big\} \\ = \Big\{ & \{c^2\}, \{a^1,c^2\}, \{a^2,c^2\}, \{a^2\}, \{a^1,a^2,c^2\}, \\ & \{a^1,a^2\}, \{c^1,c^2\}\Big\}.\end{aligned}$$

(A single representative is listed for duplicate members.) □

Sets of pseudo characters that are critical to model matching are characterized as follows.

Definition 6.5: Let $\Sigma = (A, X, x^0, f)$ be an asynchronous machine with the connected sets $\{\chi^1, \dots, \chi^m\}$, and let Σ' be a model. Let $R^d(\Sigma)$ be the diminished matrix of stable transitions of Σ and let $D(\Sigma',\Sigma)$ be given by (6.1). The *comparison matrix* of Σ relative to Σ' is an $m \times m$ matrix $S(\Sigma',\Sigma)$ with the entries

$$S_{ij}(\Sigma',\Sigma) := \begin{cases} \Pi' R^d_{ij}(\Sigma) & \text{if } (i,j) \in D(\Sigma',\Sigma), \\ \varnothing & \text{otherwise,} \end{cases}$$

$i,j = 1, 2, \dots, m$. □

An entry of the comparison matrix is a family of sets of pseudo input characters. Each member of the (i,j) entry consists of the pseudo input characters that appear in one string of $R^d_{ij}(\Sigma)$. These pseudo characters induce transitions that facilitate a move of Σ from a stable combination with a state of the connected set χ^i to a stable combination with a state of the connected set χ^j. Testing of these transitions determines the feasibility of such transitions, and hence the feasibility of model matching.

Example 6.6: Using the results of Examples 6.3 and 6.4, we obtain the comparison matrix

$$S(\Sigma',\Sigma) = \begin{pmatrix} \varnothing & \big\{\{c^2\}, \{a^1,c^2\}, \{a^2,c^2\}, \{a^2\}, \{a^1,a^2,c^2\}, \{a^1,a^2\}, \{c^1,c^2\}\big\} \\ \varnothing & \varnothing \end{pmatrix}.$$

□

Referring to Definition 2.3 of the pseudo alphabet A', recall that every pseudo character $v \in A'$ is associated with an original input character $u(v) \in A$ and with two states $z(v), z'(v)$ of Σ satisfying $z'(v) = s_a(z(v), v) \in s(z(v), u(v))$. Here, $(z(v), u(v))$ is an indeterminate pair, and $z'(v)$ is one of the possible next stable states of $(z(v), u(v))$. If $z'' \neq z'(v)$ is another possible outcome of the indeterminate pair $(z(v), u(v))$, then it is associated with a different pseudo input character $w \neq v$, where $z(w) = z(v)$, $u(w) = u(v)$, $z'(w) = z''$, and $z'(w) = s_a(z(w), w) \in s(z(w), u(w))$. In brief, each pseudo character is uniquely associated with a distinct outcome of an indeterminate pair.

Suppose now that the indeterminate pair $(z(v), u(v))$ is tested; denote by z'_{true} the outcome of the test, namely, the actual next stable state attained by Σ after applying $(z(v), u(v))$. As Σ is a deterministic machine, z'_{true} will always be the next stable state of $(z(v), u(v))$ for the present sample of Σ. Define the *true stable recursion function* $s_{true} : X \times A \to X$ by setting

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

The function s_{true} describes the outcome of testing the machine; it is not known a-priori.

Recall that the entries of the comparison matrix $S(\Sigma', \Sigma)$ indicate indeterminate transitions of the machine Σ that must be tested in order to determine whether model matching is possible. Specifically, consider a non-empty entry $S_{ij}(\Sigma', \Sigma)$. By Definition 6.5, this entry consists of a number of subsets of pseudo input characters; each subset comprises all pseudo input characters included in one of the strings that take the adjusted machine Σ_a from a stable combination with a state of the connected set χ^i to a stable combination with a state of the connected set χ^j. If the true stable transition function is compatible with all the transitions indicated by the pseudo characters of one of these subsets, then a stable transition from χ^i to χ^j is possible in Σ.

Proposition 6.7: *Let $\Sigma = (A, X, x^0, f)$ be an asynchronous machine with the diminished skeleton matrix $K^d(\Sigma)$, the adjusted stable recursion function s_a, and the true stable recursion function s_{true}. Let $\Sigma' = (A, X, x^0, s')$ be a model, and let $S(\Sigma', \Sigma)$ be the comparison matrix of Σ' relative to Σ. For a pseudo character $v \in A'$, let $(z(v), u(v)) \in X \times A$ be the indeterminate state/input pair associated with v. Then, for any pair of integers i, j for which $K^d_{ij}(\Sigma) = \Delta$, the following two statements are equivalent.*

(i) $K^d_{ij}(\Sigma)$ *turns into* 1 *after testing.*
(ii) There is a member $\theta \in S_{ij}(\Sigma', \Sigma)$ in which $s_{true}(z(v), u(v)) = s_a(z(v), v)$ for all $v \in \theta$.

Proof: Let $R^d(\Sigma)$ be the diminished matrix of stable transitions of Σ. By the definition of $K^d(\Sigma)$, the following two statements are equivalent for all $i, j \in \{1, 2, \ldots, m\}$:

(i) $K^d_{ij}(\Sigma) = 1$ after testing.
(a) $R^d_{ij}(\Sigma)$ includes a string t all of whose indeterminate transitions become true in testing, namely, $s_{true}(z(v), u(v)) = s_a(z(v), v)$ for every pseudo character v included in t.

In view of Definition 6.5, statement *(a)* is equivalent to statement *(ii)* of the Proposition. Thus, *(i)* is equivalent to *(ii)*, and our proof concludes. □

Proposition 6.7 forms the basis of a testing algorithm that determines whether or not model matching is possible with a given indeterminate machine. This algorithm depends on the comparison matrix $S(\Sigma', \Sigma)$.

6.3. Simplifying the Comparison Matrix

Often, parts of the comparison matrix $S(\Sigma', \Sigma)$ can be removed without impairing its role. Indeed, consider a non-empty entry $S_{ij}(\Sigma', \Sigma)$, and assume that it includes two members θ, θ' satisfying $\theta \subseteq \theta'$. Then, clearly, testing θ' implies testing of θ. However, if θ is tested and condition *(ii)* of Proposition 6.7 holds for θ, then $K^d_{ij}(\Sigma)$ turns into 1 after testing θ, and there is no need to test the larger set θ'. Thus, θ' can be deleted from the comparison matrix. This helps simplify the matrix.

Definition 6.8: Let $S(\Sigma', \Sigma)$ be the $m \times m$ comparison matrix of Σ relative to Σ'. For every pair of integers $i, j \in \{1, 2, \ldots, m\}$ and for every pair of distinct members $\theta, \theta' \in S_{ij}(\Sigma', \Sigma)$ for which $\theta \subseteq \theta'$, delete the member θ' from $S_{ij}(\Sigma', \Sigma)$. The resulting matrix $S^r(\Sigma', \Sigma)$ is the *reduced comparison matrix* of Σ relative to Σ'. □

Example 6.9: For the comparison matrix of Example 6.6, we obtain the reduced comparison matrix

$$S^r(\Sigma', \Sigma) = \begin{pmatrix} \varnothing & \{\{a^2\}, \{c^2\}\} \\ \varnothing & \varnothing \end{pmatrix},$$

a substantial simplification. □

Our discussion in this subsection indicates that Proposition 6.7 remains valid when the comparison matrix is replaced by the reduced comparison matrix, as follows.

Corollary 6.10: *Let $\Sigma = (A, X, x^0, f)$ be an asynchronous machine with the diminished skeleton matrix $K^d(\Sigma)$, the adjusted stable recursion function s_a, and the true stable recursion function s_{true}. Let $\Sigma' = (A, X, x^0, s')$ be a model, and let $S^r(\Sigma', \Sigma)$ be the reduced comparison matrix of Σ' relative to Σ. For a pseudo character $v \in A'$, let $(z(v), u(v)) \in X \times A$ be the indeterminate state/input pair associated with v. Then, for every pair of integers i, j for which $K^d_{ij}(\Sigma) = \Delta$, the following two statements are equivalent.*

(i) $K^d_{ij}(\Sigma) = 1$ *after testing.*
(ii) There is a member $\theta \in S^r_{ij}(\Sigma', \Sigma)$ in which $s_{true}(z(v), u(v)) = s_a(z(v), v)$ for all $v \in \theta$. □

Using Corollary 6.10, we build in the next section a testing algorithm to determine whether model matching is possible.

7. Testing Indeterminate Transitions

7.1. Testable Transitions

Our next goal is to characterize those indeterminate transitions that can be tested in the initial state. To be specific, consider an indeterminate machine $\Sigma = (A, X, x^0, f)$ with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the skeleton matrix $K(\Sigma)$, and the adjusted machine $\Sigma_a := (\tilde{A}, X, x^0, s_a)$. To demonstrate the testing process, we restrict ourselves to testing in the initial state, where the controller C drives Σ in fundamental mode operation along round trips that start and end at the initial state x^0. A similar process can be used to test Σ in any stable state.

Testing in the initial state can involve only states of the following set, as these are the only states with a guaranteed route back to the initial state

$$\rho^0 := \{x^i \in X \,|\, K_{ij}(\Sigma) = 1, \text{ where } x^0 = x^j \text{ is the initial state}\}. \quad (7.1)$$

Example 7.1: For the machine Σ of Example 2.4 with the initial state $x^0 = x^1$, a direct examination of Example 4.9 shows that $\rho^0 = X$, the entire state set of Σ. □

Note that ρ^0 cannot be empty, since we always have $K_{jj}(\Sigma) = 1$ and whence $x^0 = x^j \in \rho^0$. However, if ρ^0 includes only the initial state, then, clearly, testing in the initial state is not meaningful.

Now, in order to be able to test an indeterminate pair (x, u), it must be possible to reach the state x from the initial state x^0, at least for some outcomes of indeterminate transitions. Thus, x must be a member of the set

$$S^0 := \{x^i \in X \,|\, K_{ji}(\Sigma) \neq 0, \text{ where } x^0 = x^j \text{ is the initial state}\}. \quad (7.2)$$

Note that $S^0 \neq \varnothing$ since $K_{jj}(\Sigma) = 1$.

Example 7.2: For the machine Σ of Example 2.4, it follows by Example 4.9 that $S^0 = X$, the entire state set. □

Having reached the state x, it must always be possible to drive Σ back to the initial state x^0 from any outcome of (x, u). Thus, letting s be the stable recursion function of Σ, we conclude that any testable pair (x, u) must be a member of the set

$$T(\Sigma, x^0) := \{(x, u) \in S^0 \times A \,|\, (x, u) \text{ is a valid pair and } s(x, u) \subseteq \rho^0\}. \quad (7.3)$$

Example 7.3: For the machine Σ of Example 2.4, it follows by Examples 7.1 and 7.2 that $T(\Sigma, x^0)$ consists of all valid pairs of Σ. □

We summarize our discussion thus far in the following statement.

Proposition 7.4: *Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with stable recursion function s, and let S^0 and ρ^0 be given by (7.2) and (7.1), respectively. Then, an indeterminate pair of Σ can be tested in the initial state only if it belongs to the set $T(\Sigma, x^0)$ of (7.3).* □

Let $\Pi_x : X \times A \to X : \Pi_x(x, u) := x$ be the projection onto the state. Then, by Proposition 7.4, testing in the initial state can involve only states of the set

$$\Pi_x T(\Sigma, x^0).$$

Some members of this set may not be reachable in a particular sample of Σ, since the set S^0 of (7.2) includes outcomes of indeterminate transitions. Equivalently, some members of $T(\Sigma, x^0)$ may not be available for testing in a particular sample of Σ. A brief examination shows that all pairs whose testing can be guaranteed a-priori are characterized by the following.

Proposition 7.5: *Let Σ be an indeterminate asynchronous machine with skeleton matrix $K(\Sigma)$ and initial state $x^0 = x^j$, and let $T(\Sigma, x^0)$ be given by (7.3). Then, the following two statements are equivalent for an indeterminate pair (x, u) of Σ.*

(i) (x, u) can always be tested, irrespective of the outcomes of any indeterminate transitions.
(ii) (x, u) is a member of the set

$$\tau(\Sigma, x^0) := \{(x^i, u) \in T(\Sigma, x^0) \,|\, K_{ji}(\Sigma) = 1\}. \square \quad (7.4)$$

In view of Proposition 7.5, we refer to $\tau(\Sigma, x^0)$ as the set of *certainly testable pairs*.

Example 7.6: For the machine Σ of Example 2.4, it follows from Example 4.9 that $\tau(\Sigma, x^0)$ consists of all valid pairs within $\{x^1, x^2, x^3\} \times A$. □

Let $\{(z^1, u^1), \ldots, (z^r, u^r)\}$ be the set of all indeterminate pairs of the machine Σ, and let A^i be the set of pseudo characters associated with the indeterminate pair (z^i, u^i), as described in (2.3). Denote by $A'(\Sigma, x^0)$ the set of all pseudo characters associated with the set $T(\Sigma, x^0)$ of (7.3), namely,

$$A'(\Sigma, x^0) = \bigcup_{\{i | (z^i, u^i) \in T(\Sigma, x^0)\}} A^i. \quad (7.5)$$

By Proposition 7.4, pseudo characters outside this set cannot be tested in the initial state. Similarly,

$$A'_c(\Sigma, x^0) = \bigcup_{\{i | (z^i, u^i) \in \tau(\Sigma, x^0)\}} A^i \quad (7.6)$$

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

Table 4. The family T of complete sets.

Entire entry	The family T of complete sets (only a few terms are listed in each case)	$\bigcup_{t \in T} \Pi' t$
$(R^a(\Sigma, 1))_{11}$	$\{\cdots\}$	$\varnothing$
$(R^a(\Sigma, 1))_{11}^{(2)}$	$\{\cdots\}$	$\varnothing$
$(R^a(\Sigma, 1))_{11}^{(3)}$	$\{a\gamma_1, a\gamma_2, a\gamma_3\}$	$\{c^1, c^2\}$
$(R^a(\Sigma, 1)_{11}^{(4)}$	$\{(R^a(\Sigma, 1))_{11}^{(3)}, a\gamma_1 b, a\gamma_2 b, a\gamma_3 b, a\gamma_1 c, a\gamma_2 c, a\gamma_3 c, ba\gamma_1, \cdots, ca\gamma_1, \cdots\}$	$\{c^1, c^2\}$
$(R^a(\Sigma, 1)_{11}^{(5)}$	$\{(R^a(\Sigma, 1)_{11}^{(4)}, bca\gamma_1, bca\gamma_2, bca\gamma_3, cba\gamma_1, \cdots, ad\gamma_4, \cdots\}$	$\{a^1, a^2, c^1, c^2\}$

of Σ: samples that have favorable outcomes of appropriate indeterminate transitions.

Example 7.7: For the machine Σ of Example 2.4, the initial state is $x^0 = x^1$. Considering Examples 7.3 and 7.6, it follows that

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

namely, all pseudo characters appear in both sets in this case. □

The next statement provides a bound on the complexity of testing all indeterminate transitions that can be tested in the initial state. It also provides a basis for selecting testing strings by specifying the family T of the statement. This family consists of complete strings that take Σ through round trips from the initial state back to the initial state, passing through every indeterminate pair that is certainly testable.

Lemma 7.8: *Let* $\Sigma = (A, X, x^0, f)$ *be an indeterminate asynchronous machine with the state set* $\{x^1, \ldots, x^n\}$*, the initial state* $x^0 = x^\alpha$*, and the one-step matrix of stable transitions* $R^a(\Sigma, 1)$*, and let* $A'_c(\Sigma, x^0)$ *be the set of certainly testable pseudo characters given by (7.6). Then,*

(i) There is an integer $\mu > 0$ *such that the entry* $(R^a(\Sigma, 1))_{\alpha\alpha}^{(\mu)}$ *includes a family* T *of complete strings satisfying* $A'_c(\Sigma, x^0) \subseteq \bigcup_{t \in T} \Pi' t$*, and*
(ii) $\mu \leq 2n - 1$.

Proof: Let $\{(z^1, u^1), \ldots, (z^r, u^r)\}$ be the set of indeterminate pairs of the machine Σ. In view of Proposition 7.5, the indeterminate pair (z^i, u^i) is certainly testable if and only if $(z^i, u^i) \in \tau(\Sigma, x^0)$. If $\tau(\Sigma, x^0)$ includes no indeterminate pairs, then *(i)* and *(ii)* are valid with T being the empty set and $\mu = 0$. Otherwise, there is an integer $i \in \{1, 2, \ldots, r\}$ such that $(z^i, u^i) \in \tau(\Sigma, x^0)$. Then, by (7.4), there is a complete set $v^i \in (\tilde{A})^+$ that takes Σ_a from the initial state x^0 to a stable combination with the state z^i. In view of Proposition 3.3, we can select v^i to have $n - 1$ or fewer steps.

At the state z^i, apply the character u^i and denote by $z^{i,j}$ the resulting next stable state. By (7.4), it follows that there is a complete set $w^{i,j} \in (\tilde{A})^+$ that takes Σ_a from a stable combination with $z^{i,j}$ to a stable combination with the initial state $x^0 = x^\alpha$. Applying again Proposition 3.3, we conclude that $w^{i,j}$ can be taken to have $n - 1$ or fewer steps. Thus, the input set $\gamma := v^i u^i w^{i,j}$ induces a round trip from x^0 back to x^0 passing through the indeterminate pair (z^i, u^i), and it satisfies $|\gamma| \leq 2(n - 1) + 1 = 2n - 1$. As this is valid for all $i = 1, \ldots, r$ and all $j = 1, 2, \ldots, n(i)$, our proof concludes. □

Example 7.9: Consider the machine Σ of Example 2.4. In our case, $x^0 = x^1$, so that $\alpha = 1$. Using Example 4.2, we examine entry $(1, 1)$ of consecutive powers of $R^a(\Sigma, 1)$ in Table 4, where $\gamma_1 := c^1 + c^2 b$, $\gamma_2 := (c^1 + c^2)b$, $\gamma_3 := c^1 c + c^2 b$, and $\gamma_4 := a^1 \gamma_1 + a^2 b$ are complete sets. As we can see, all pseudo characters of Σ appear in the last row of the table. This demonstrates Lemma 7.8*(ii)*, since $\mu = 5$ and $n = 4$ here. □

Remark 7.10: Let Σ be an indeterminate asynchronous machine. Referring to (7.5) and (7.4), consider the case when $A'(\Sigma, x^0) \neq A'_c(\Sigma, x^0)$. Then, members of the difference set $A'(\Sigma, x^0) \setminus A'_c(\Sigma, x^0)$ can be tested only under particular outcomes of some indeterminate transitions of Σ. In such case, it is convenient to employ multiple rounds of testing.

To this end, let Σ^1 be the asynchronous machine obtained after the outcomes of initial testing were incorporated as determinate transitions into Σ. The machine Σ^1 is a new indeterminate machine with a new set $A'_c(\Sigma^1, x^0)$ of pseudo characters associated with certainly testable transitions. After testing Σ^1 and recording the outcomes of all tested indeterminate transitions, we obtain the asynchronous machine Σ^2, and so on.

This process continues until no more indeterminate transitions can be reached for testing. Let Σ^i be the asynchronous machine obtained after incorporating the outcomes of indeterminate transitions tested in the i-th round of testing. Then, testing terminates when the diminished skeleton matrix stops changing, namely, when $K^d(\Sigma^i) = K^d(\Sigma^{i+1})$. □

7.2. An Algorithm for Testing

We are ready now to present a mechanism by which the controller C of Fig. 1 can automatically test indeterminate transitions of the controlled machine Σ. It goes without saying that only indeterminate transitions that are relevant to the model matching problem at hand need to be tested. As before, we restrict ourselves to testing in the initial state, namely, to testing that starts and ends at the initial state of Σ and is performed when the controller C is turned on. The restriction to testing in the initial state comes only to shorten the presentation; testing with similar protocol can be performed in any stable state of the controlled machine Σ.

During testing, the controller follows the algorithm below. This algorithm directs the controller to take the closed loop machine Σ_c through a chain of transient transitions that starts and ends at the initial state x^0 of Σ. In brief terms, the algorithm can be described as follows. First, the algorithm finds all indeterminate pairs of Σ that can potentially be tested (see (7.3)); their associated pseudo characters are given by (7.5). If none of these pseudo characters appear in the reduced comparison matrix $S^r(\Sigma', \Sigma)$, then the algorithm terminates, since testing cannot resolve any meaningful indeterminacies. Otherwise, the algorithm proceeds to apply to the machine Σ strings of the set T of Lemma 7.8 and records the outcome of any indeterminate transitions of Σ encountered. These outcomes are then used to update the skeleton matrix of Σ, potentially replacing some entries of Δ by entries of 0 or 1. The process repeats until all strings of T have been exhausted, or until the existence (or non-existence) of a model matching controller can be ascertained.

The entire process of testing is implemented by the controller C of Fig. 1 as a string of transient transitions of the closed loop machine Σ_c. As transients occur very quickly in asynchronous machines (ideally, in zero time), this testing process does not interfere with user experience.

Algorithm 7.11 (Testing in the Initial State): *Let $\Sigma = (A, X, x^0, f)$ be an indeterminate asynchronous machine with adjusted machine $\Sigma_a = (\tilde{A}, X, x^0, s_a)$, state set $X = \{x^1, \dots, x^n\}$, initial state $x^0 = x^\alpha$, family of connected sets $\Xi = \{\chi^1, \dots, \chi^m\}$, true stable recursion function s_{true}, initial connected set χ^0, one-step matrix of stable transitions $R^a(\Sigma, 1)$, and diminished skeleton matrix $K^d(\Sigma)$. Let $\Sigma' = (A, X, x^0, s')$ be a model with diminished skeleton matrix $K^d(\Sigma', \Sigma)$ relative to Σ, and let $S^r(\Sigma', \Sigma)$ be the reduced comparison matrix.*

Step 0. *Set $\beta := 0$ and $K^d(\Sigma, 0) := K^d(\Sigma)$.*

Step 1. *Let $A'(\Sigma, x^0)$ be given by (7.5). If no pseudo characters of $A'(\Sigma, x^0)$ appear in the matrix $S^r(\Sigma', \Sigma)$, then testing of indeterminate transitions in the initial state is not meaningful; go to Step 9.*

Step 2. *Let T and μ be as given in Lemma 7.8, and let $t_1, t_2, \dots, t_q$ be the complete sets of strings included in T. Define the subsets of pseudo characters $\theta_i := \Pi' t_i, i = 1, 2, \dots, q$. Replace β by $\beta + 1$, set $j := 1$, and set $K^d(\Sigma, \beta) := K^d(\Sigma, \beta - 1)$.*

Step 3. *If θ_j has no characters in common with the matrix $S^r(\Sigma', \Sigma)$, then go to Step 7.*

Step 4. *Apply the complete set of strings t_j to the machine Σ in fundamental mode operation, as follows:*
After each character, wait until Σ has reached its next stable state and record it.
Based on the stable state reached by Σ, select the subsequent input character v; if v is a pseudo character, apply to Σ the original character $u(v) \in A$ associated with v.
Continue in this character-by-character manner until Σ returns to its initial state x^0.

Step 5. *Using the results of Step 4, partition the set θ_j into two disjoint subsets θ_j^+ and θ_j^-, where θ_j^+ consists of all pseudo input characters $v \in \theta_j$ for which $s_{true}(z(v), u(v)) = s_a(z(v), v)$, and $\theta_j^- := \theta_j \setminus \theta_j^+$ (the set difference).*

Step 6. *Perform sub-steps (a) to (d) below for every pair of integers $p, \ell \in \{1, \dots, m\}$, where m is the number of connected sets in the family Ξ:*

(a) If there is a member θ of $S^r_{p\ell}(\Sigma', \Sigma)$ satisfying $\theta \subseteq \theta_j^+$, then assign $K^d_{p\ell}(\Sigma, \beta) := 1$, and replace $S^r_{p\ell}(\Sigma', \Sigma)$ by the empty set.

(b) Replace every member θ of $S^r_{p\ell}(\Sigma', \Sigma)$ by the difference set $\theta \setminus \theta_j^+$.

(c) Remove from $S^r_{p\ell}(\Sigma', \Sigma)$ every member that is not disjoint with θ_j^-. If this turns $S^r_{p\ell}(\Sigma', \Sigma)$ into the empty set, assign $K^d_{p\ell}(\Sigma, \beta) := 0$; then, if $K^d_{p\ell}(\Sigma', \Sigma) = 1$, set $K^d(\Sigma) := K^d(\Sigma, \beta)$ and go to Step 9 (model matching is not possible).

(d) If $K^d(\Sigma, \beta) \geq K^d(\Sigma', \Sigma)$, then set $K^d(\Sigma) := K^d(\Sigma, \beta)$ and go to Step 9 (model matching is possible).

Step 7. *If $j < q$, replace j by $j + 1$ and return to Step 3.*

Step 8. *If $K^d(\Sigma, \beta) \neq K^d(\Sigma, \beta - 1)$, then recalculate the family of connected sets Ξ of Σ, the one-step matrix of stable transitions $R^a(\Sigma, 1)$, the diminished skeleton matrix $K^d(\Sigma)$, the diminished skeleton matrix $K^d(\Sigma', \Sigma)$, and the reduced comparison matrix $S^r(\Sigma', \Sigma)$. Set $K^d(\Sigma, \beta) := K^d(\Sigma)$ and Return to Step 1.*

Step 9. Output the matrices $K^d(\Sigma)$ and $K^d(\Sigma', \Sigma)$, and terminate the algorithm. □

Remark 7.12: To shorten the presentation in this paper, we restrict testing to the initial state of the tested machine Σ. The same principles can be used to test indeterminate transitions of Σ in any stable state x of Σ by designing the controller C to drive Σ in round trips from x back to x. This would encompass all testing of indeterminate transitions of Σ that is possible in fundamental mode operation. □

According to the next statement, Algorithm 7.11 can be implemented within the controller C of Fig. 1 to test and determine automatically whether or not a desired model can be matched. This determination is made before the closed loop machine Σ_c receives any external operational commands. If the desired model turns out to be unmatchable, the model can be altered to a compatible model before operational use of the closed loop machine commences.

Theorem 7.13: *Let $\Sigma = (A, X, x^0, f)$ be an asynchronous machine, let $\Sigma' = (A, X, x^0, s')$ be a model, and let $K^d(\Sigma)$ and $K^d(\Sigma', \Sigma)$ be the outcomes of Algorithm 7.11. Then, the following are valid:*

(i) Algorithm 7.11 can be implemented by a state feedback controller in fundamental mode operation.

(ii) Statements (a) and (b) are equivalent:

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

(b) $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$.

(iii) If there is a pair of integers i, j for which $K^d_{ij}(\Sigma) = 0$ while $K^d_{ij}(\Sigma', \Sigma) = 1$, then there is no controller satisfying $\Sigma_c = \Sigma'$ in fundamental mode operation.

(iv) If there is a pair of integers i, j for which $K^d_{ij}(\Sigma) = \Delta$ while $K^d_{ij}(\Sigma', \Sigma) = 1$, then testing in the initial state cannot determine whether there exists a controller C for which $\Sigma_c = \Sigma'$ in fundamental mode operation.

Proof: (i) Refer to Step 4 of Algorithm 7.11. In the notation of the algorithm, consider a complete set of strings $t_j \in T = \{t_1, t_2, \ldots, t_q\}$, and let $v \in \tilde{A}$ be a character of t_j. Denote by $u(v) \in A$ the original input character associated with v. By Convention 6.2, all state/input pairs of Σ are detectable. Thus, a state feedback controller can determine when the machine Σ has reached its next stable state in response to the input character $u(v)$. Consequently, for each $j = 1, 2, \ldots, q$, a state feedback controller C'_j that implements the complete set of strings t_j can be built using the construction of [13, proof of Theorem 5.1]. Denote by C_j the state feedback controller obtained by adding to C'_j memory to record the outcomes of all encountered indeterminate transitions.

Now, combine the controllers $C_1, \ldots, C_q$ into a single sequential controller C_T by activating C_{j+1} when C_j detects a return of Σ to the initial state x^0, $j = 1, 2, \ldots, q-1$. Then, the controller C_T implements Step 4 of Algorithm 7.11 in fundamental mode operation, and we conclude that *(i)* is valid.

Statements *(ii)*, *(iii)*, and *(iv)* follow from Corollary 6.10, Theorem 5.4, and Remark 7.10. This completes our proof. □

Table 5. The true stable recursion function.

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

Remark 7.14: Once the controller has completed Algorithm 7.11 and has determined that model matching is possible, it activates a model matching component that is constructed as described in [13, proof of Theorem 5.1]. □

Example 7.15: Consider the problem of designing a controller C for the indeterminate machine Σ of Example 2.4 to match the model Σ' of Example 5.2. Assume that the true stable recursion function s_{true} of Σ (which is to be obtained from the testing described below) is as shown in Table 5.

To determine whether there is a controller C that solves the requisite model matching problem with this sample of Σ, we use Algorithm 7.11 and Theorem 7.13. According to Example 6.9, we must test only the entry $S^r_{12}(\Sigma', \Sigma) = \{\{a^2\}, \{c^2\}\}$. Recall that the initial condition of the machine Σ is the state $x^0 = x^1$, which is included in the connected set $\chi^1 = \{x^1, x^2, x^3\}$; the one-step adjusted matrix of stable transition $R^a(\Sigma, 1)$ is given in Example 4.2. We perform testing in the initial state x^1 of Σ using Algorithm 7.11.

According to Example 7.9, we have

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

here, we listed only one representative from each class of complete sets with the same pseudo characters. According to (7.5), we have

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

Step 0. Set $\beta := 0$ and $K^d(\Sigma, 0) := K^d(\Sigma)$, where $K^d(\Sigma)$ is given in Example 4.16.

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

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

Step 3: Recall from Example 6.9 that $S^r_{12}(\Sigma', \Sigma) = \{\{a^2\}, \{c^2\}\}$. As θ_1 has the pseudo character c^2 in common with $S^r_{12}(\Sigma', \Sigma)$, proceed to the next step.

Step 4: In view of Example 2.4, we have $u(c^1) = c$ and $u(c^2) = c$, namely, c is the real character that corresponds to the pseudo characters c^1 and c^2. Therefore, based on the complete set $t_1 = a(c^1 + c^2 b)$, drive the machine Σ as follows:
At the initial state $x^0 = x^1$, apply the input character a to reach the stable state x^2.
Upon reaching x^2, apply to Σ the input character $c = u(c^1) = u(c^2)$, which activates an indeterminate transition of Σ.
According to Table 5, the next stable state of Σ turns out to be x^4.

Step 5: Considering the adjusted stable recursion function s_a of Table 2, we conclude that the true transition corresponds to the pseudo character c^2. Therefore,

$$\theta_1^+ = \{c^2\}, \theta_1^- = \{c^1\}.$$

Step 6: *(a)* Since $\theta_1^+ = \{c^2\}$, and $\{c^2\}$ is a member of $S^r_{12}(\Sigma', \Sigma)$, set $K^d_{12}(\Sigma) = 1$ and $S^r_{12}(\Sigma', \Sigma) := \varnothing$. Then, (d) is valid, and model matching is possible. The algorithm terminates.

Further, in this case, since $K^d_{12}(\Sigma) = 1$, it follows that all states of the current sample of the machine Σ are reachable from each other. Consequently, the state set of this sample of Σ consists of a single connected set, and all relevant matrices become scalars. Recalculating, we obtain $K^d(\Sigma) = 1$ and $K^d(\Sigma', \Sigma) = 1$, so that $K^d(\Sigma) \geq K^d(\Sigma', \Sigma)$. Thus, Theorem 7.13*(ii)* guarantees the existence of a state feedback controller C that operates in fundamental mode and satisfies $\Sigma_c = \Sigma'$. This controller can be built as described in [13, proof of Theorem 5.1]. □

8. Conclusion

The paper presents a methodology for controlling indeterminate asynchronous machines. Control is performed by an adaptive controller that automatically performs three functions: *(i)* it tests critical indeterminate transitions of the controlled machine; *(ii)* based on the results of testing, it determines whether model matching to a specified model is possible; and *(iii)* if model matching is possible, it drives the controlled machine so as to match the specified model.

The process of testing indeterminate transitions of the controlled machine is performed during transients of the closed loop machine Σ_c; as transients of asynchronous machines pass quickly (ideally, in zero time), the testing process does not interfere with user experience. Once testing is complete, the controller adjusts its function to the test's outcome, thus forming an adaptation process.

The discussion in this paper is restricted to adaptive state feedback controllers. The design of adaptive output feedback controllers for asynchronous sequential machines will be considered in a separate report.

Acknowledgement

The work of J.-M. Yang was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2012-0002849).

References

1. Barrett G, Lafortune S. Bisimulation, the supervisory control problem, and strong model matching for finite state machines. *Discrete Event Dyn Syst: Theory Appl*, 1998; 8(4): 377–429.
2. Dibenedetto MD, Saldanha A, Sangiovanni-Vincentelli A. Model matching for finite state machines. Proceedings of the IEEE Conference on Decision and Control, Vol. 3, pp. 3117–3124, 1994.
3. Geng XJ, Hammer J. Asynchronous sequential machines: input/output control. Proceedings of the 12th Mediterranean Conference on Control and Automation, Kusadasi, Turkey, June 2004.
4. Geng XJ, Hammer J. Input/output control of asynchronous sequential machines. *IEEE Trans Automat Control*, 2005; 50(12): 1956–1970.
5. Hammer J. On some control problems in molecular biology. Proceedings of the IEEE Conference on Decision and Control, pp. 4098–4103, December 1994.
6. Hammer J. On the modeling and control of biological signal chains. Proceedings of the IEEE Conference on Decision and Control, pp. 3747–3752, December 1995.
7. Hammer J. On the corrective control of sequential machines. *Int J Control*, 1996; 65(2): 249–276.
8. Hammer J. On the control of incompletely described sequential machines. *Int J Control*, 1996; 63(6): 1005–1028.
9. Hammer J. On the control of sequential machines with disturbances. *Int J Control*, 1997; 67(3): 307–331.
10. Kohavi Z. Switching and Finite Automata Theory, McGraw-Hill, New York, NY, 1978.
11. Kumar R, Nelvagal S, Marcus SI. A discrete event systems approach for protocol conversion. *Discrete Event Dyn Syst: Theory Appl*, 1997; 7(3): 295–315.
12. Murphy TE, Geng XJ, Hammer J. Controlling races in asynchronous sequential machines. Proceeding of the IFAC World Congress, Barcelona, July 2002.

13. Murphy TE, Geng X, Hammer J. On the control of asynchronous machines with races. *IEEE Trans Autom Control*, 2003; 48(6): 1073–1081.
14. Ramadge PJG, Wonham WM. Supervisory control of a class of discrete event processes. *SIAM J Control Optim*, 1987; 25(1): 206–230.
15. Shieh M-D, Wey C-L, Fisher PD. Fault effects in asynchronous sequential logic circuits. *IEE Proc-E*, 1993; 140(6): 327–332.
16. Thistle JG, Wonham WM. Control of infinite behavior of finite automata. *SIAM J Control Optim*, 1994; 32(4): 1075–1097.
17. Venkatraman N, Hammer J. Stable realizations of asynchronous sequential machines with infinite cycles. Proceedings of the 2006 Asian Control Conference, Bali, Indonesia, pp. 45–51, 2006.
18. Venkatraman N, Hammer J. Controllers for asynchronous machines with infinite cycles. Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, pp. 1002–1007, 2006.
19. Venkatraman N, Hammer J. On the control of asynchronous sequential machines with infinite cycles. *Int J Control*, 2006; 79(7): 764–785.
20. Yang J-M, Hammer J. Counteracting the effects of adversarial inputs on asynchronous sequential machines. Proceedings of the IFAC World Congress, Seoul, Korea, pp. 1432–1437, July 2008.
21. Yang J-M, Hammer J. State feedback control of asynchronous sequential machines with adversarial inputs. *Int J Control*, 2008; 81(12): 1910–1929.
22. Yevtushenko N, Villa T, Brayton R, Petrenko A, Sangiovanni-Vincentelli A. Compositionally progressive solutions of synchronous FSM equations. *Discrete Event Dyn Syst: Theory Appl*, 2008; 18(1): 51–89.