INTERNATIONAL JOURNAL OF GENERAL SYSTEMS, 2016
VOL. 45, NOS. 7–8, 830–863
http://dx.doi.org/10.1080/03081079.2016.1158513

Taylor & Francis
Taylor & Francis Group

Static state feedback control of asynchronous sequential machines

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

[a]School of Electronics Engineering, Kyungpook National University, Daegu, Republic of Korea; [b]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA

ABSTRACT

Static controllers, which consist of logical gates with no memory elements, form the simplest class of controllers for asynchronous sequential machines. This paper presents necessary and sufficient conditions for the existence of static state feedback controllers that control a given asynchronous sequential machine so as to match a specified model. The process of designing static state feedback controllers is described.

ARTICLE HISTORY
Received 16 December 2014
Accepted 14 February 2016

KEYWORDS
Asynchronous sequential machines; state feedback; non-linear control

1. Introduction

Asynchronous sequential machines, or – as they are often called – clockless logic circuits, play a critical role in the construction of high-speed computing systems and in the modelling of signalling chains in molecular biology (Hammer 1994). The use of asynchronous sequential machines in advanced computers, in the design of new-generation microprocessors, and in the implementation of parallel and distributed digital systems has become more and more prevalent in recent years, as performance demands on computing systems have continued to grow. The use of asynchronous sequential machines has also expanded into digital communications, as asynchronous digital communication networks have started to take hold (Sparsø and Furber 2001; Martin and Nyström 2006; Tinder 2009; and others).

The presence of asynchronous sequential machines in broader and broader application areas has made it more and more common for engineers to face the task of dealing with asynchronous sequential machines that exhibit undesirable behaviour. When encountering an asynchronous sequential machine that exhibits undesirable behaviour, one has to choose between two options: (i) replace the faulty machine with a newly designed one; or (ii) attempt to correct the behaviour of the faulty machine. The first option, namely that of replacing the faulty machine, is not always practical. For example, consider an asynchronous sequential machine representing a biological signalling chain that has developed defects as a result of senescence or viral infection. It goes without saying that such a machine cannot be replaced by a newly designed one, as the machine forms an inseparable part of an organism. Similarly, asynchronous machines that form parts of inaccessible systems,

CONTACT Jacob Hammer hammer@mst.ufl.edu

such as devices implanted in the human body or devices placed on remote platform in space, cannot be readily replaced.

In such cases, one must attempt to correct the behaviour of a faulty asynchronous machine, rather than replace it. The behaviour of a faulty asynchronous sequential machine can often be corrected by an external controller that monitors the machine and applies commands that steer the machine towards acceptable behaviour. Such controllers can be designed through control theoretic methodologies; they are referred to as *corrective controllers*. Corrective controllers for asynchronous computing machines can often be implemented remotely by the use of software patches; corrective controllers for biological systems can be inserted through the use of viral vectors. As a result, corrective controllers emerge as an important tool in a wide range of critical applications.

The use of control theoretic techniques to overcome deficiencies in the operation of asynchronous sequential machines has been discussed in Murphy, Geng, and Hammer (2002, 2003), where dynamic state feedback controllers for asynchronous machines are investigated; in Geng and Hammer (2005), where dynamic output feedback controllers for asynchronous machines are examined; in Venkatraman and Hammer (2006b, 2006c), where dynamic state feedback controllers are used to eliminate the effects of infinite cycles on asynchronous sequential machines; in Yang and Hammer (2008, 2010), where dynamic feedback controllers are used to counteract adversarial interventions; in Peng and Hammer (2010, 2012), where dynamic output feedback controllers are used to overcome indeterminacy in non-deterministic asynchronous machines; in Yang and Kwak (2010), where industrial applications of dynamic feedback controllers for asynchronous sequential machines are demonstrated; and in Yang (2011), where the reduction of asynchronous machines is discussed.

The controllers employed in these publications are all dynamic feedback controllers, namely they are formed by asynchronous sequential machines that include memory elements. In the present paper, we direct our attention to the simplest class of controllers – static state feedback controllers. Static controllers consist only of logical gates and require no memory elements; as a result, they are simpler to implement. The present paper characterizes the conditions under which static state feedback controllers can be used to achieve a prescribed control objective, thus characterizing cases in which more complex controllers can be avoided. As one might expect, the control objectives that can be met with static state feedback controllers are somewhat more restrictive than those that can be met with dynamic state feedback controllers. Nonetheless, as we show, static state feedback controllers can be used to accommodate a range of control objectives.

In technical terms, a static state feedback controller is represented by a function that assigns an input character to each state of the controlled machine. This contrasts with a dynamic controller, which is represented by an asynchronous sequential machine that generates strings of input characters in response to state transitions of the controlled machine.

To be specific, recall that an input/state asynchronous sequential machine Σ is characterized by a triplet (A, X, f), where A is the input alphabet (or the input set), X is the state set, and $f : X \times A \to X$ is the *recursion function* of Σ. In operation, the machine Σ starts from a given initial state x_0 and is driven by a sequence of input characters $u_0, u_1, u_2, \ldots \in A$. In response to this input sequence, the machine embarks from its initial state on a sequence of states $x_1, x_2, \ldots \in X$ according to the recursion

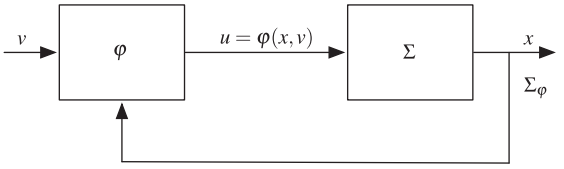

Figure 1. The static state feedback configuration.

$$\Sigma: \quad x_{k+1} = f(x_k, u_k), \quad k = 0, 1, 2, \ldots \tag{1}$$

As depicted in Figure 1, a static state feedback controller for the machine Σ is represented by a function $\varphi : X \times A \to A$ that generates the current input character u_k of Σ from the current state x_k of Σ and the current external input character v_k according to the relation

$$u_k = \varphi(x_k, v_k), \quad k = 0, 1, 2, \ldots$$

When the feedback function φ is combined with Σ, we obtain the closed-loop machine Σ_φ depicted in Figure 1 and described by the recursion

$$\Sigma_\varphi: \quad x_{k+1} = f(x_k, \varphi(x_k, v_k)), \quad k = 0, 1, 2, \ldots \tag{2}$$

In practice, static state feedback functions are implemented by logical gates; no memory elements are used.

In this paper, we concentrate on characterizing the capabilities of static state feedback controllers, when used as tools to achieve model matching of asynchronous sequential machines. In formal terms, we address the following issue.

Problem 1: Given a pair of asynchronous machines $\Sigma = (A, X, f)$ and $\Sigma' = (A, X, f')$ with the same state set X and the same input alphabet A, find necessary and sufficient conditions for the existence of a state feedback function $\varphi : X \times A \to A$ for which $\Sigma_\varphi = \Sigma'$. If such a function exists, describe its construction.

In Problem 1, the system Σ' serves as the model that must be matched by the closed-loop machine Σ_φ. Necessary and sufficient conditions under which model matching of asynchronous machines can be achieved by static state feedback are derived in Sections 4 and 5.

A few words are in place regarding the equality $\Sigma_\varphi = \Sigma'$ listed in Problem 1. Recall that an asynchronous machine can be in one of two kinds of states: a stable state – a state in which the machine lingers until a change occurs in its input; or a transient state – a state through which the machine passes quickly (ideally, in zero time) on its way from one stable state to another. Users are aware only of stable states, since transient states are traversed by the machine very quickly. Accordingly, the equality $\Sigma_\varphi = \Sigma'$ of Problem 1 refers to stable transitions, namely to transitions between stable states. It means that Σ_φ and Σ' have the same stable states and the same transitions between stable states. A formal discussion of this point is provided in Subsection 2.1.

Another issue that is important to the operation of asynchronous machines relates to the fact that the inputs of an asynchronous machine must be kept constant as long as the machine is in transition; input characters are allowed to change only when an asynchronous machine is in a stable state. The reason for this is rather simple. Due to

speed and asynchrony, it is not possible to aim an input change at a particular transient state; an asynchronous machine in transition moves through its transient states so quickly that it is not possible to predict at what state the machine will be when an input change actually occurs. As a result, changing an input value while an asynchronous machine is in transition may result in an unpredictable outcome. This need to restrict changes in the input of an asynchronous machine to times at which the machine is in a stable state leads to the notion of *fundamental mode operation*, an operating policy whereby input characters of an asynchronous machine can change only when the machine is resting at a stable state (e.g. Kohavi 1978).

The main difference between the impact of a static state feedback controller and that of a dynamic state feedback controller on the controlled machine $\Sigma = (A, X, f)$ can be visualized in rather simple terms. A static state feedback controller, being described by a function $\varphi : X \times A \to A$, always creates the same input character for Σ in response to a pair $(x, v) \in X \times A$; on the other hand, a dynamic controller may create a different input character in response to the same pair (x, v), depending on the history of the closed-loop machine, since different histories may place the controller in different states of its own when encountering the pair (x, v). This fundamental difference between these two types of controllers makes the conditions for model matching by static state feedback controllers somewhat more restrictive, when compared with the conditions for model matching by dynamic state feedback controllers. However, the practical simplicity of implementing static controllers gives them an edge, whenever they are applicable.

In this paper, we consider two types of model matching by static state feedback controllers. The first type, considered in Section 4 below, examines the case where the input alphabet of the closed-loop machine Σ_φ can be different from the input alphabet of the controlled machine Σ; in a sense, a 'natural' input alphabet is used for Σ_φ instead of the original input alphabet A of Σ. A lookup-table is then utilized to translate between characters of the two alphabets. If it is permitted to use such a natural input alphabet for controlling Σ_φ, then the necessary and sufficient condition for model matching becomes exceedingly simple; it can be expressed as an inequality between two numerical matrices of zeros and ones. In its nature, this condition is similar to the necessary and sufficient condition for model matching by dynamic state feedback controllers derived in Murphy, Geng, and Hammer (2002, 2003), but the numerical matrices involved in the present case are different. The construction of static state feedback controllers that achieve model matching via this methodology is described in Section 4.

In traditional model matching, the input alphabet of the closed-loop machine Σ_φ must be equal to the common input alphabet of the controlled machine Σ and the model Σ'. Necessary and sufficient conditions for the existence of static state feedback controllers that achieve traditional model matching of asynchronous sequential machines are derived in Section 5. Here, in addition to the numerical matrix inequality discussed in the previous paragraph, a further condition appears. This condition comes to guarantee that the feedback function can be implemented in a consistent manner. The construction of appropriate static feedback controllers is also described in Section 5.

Model matching has been widely studied in the control theoretic literature. In particular, model matching for sequential machines has been examined within an automata theoretic framework in Hammer (1994, 1996), Barrett and Lafortune (1998), Yevtushenko et al. (2008), and others. Model matching was also investigated within the theory of supervisory

control (see, e.g., Thistle and Wonham (1994), Kumar, Nelvagal, and Marcus (1997), and many others), as well as in a large variety of other frameworks and contexts. The studies mentioned in this paragraph do not address specialized issues that relate to the operation of asynchronous sequential machines, such as the distinction between stable and transient states or the notion of fundamental mode operation.

The present paper is organized as follows. Section 2 explores the notion of fundamental mode operation for asynchronous machines with static state feedback controllers. Section 3 examines the implementation of static state feedback controllers and its implications on the conditions under which such controllers can achieve model matching. Necessary and sufficient conditions for model matching with static state feedback controllers are derived in Sections 4 and 5, where Section 4 concentrates on model matching with an expanded input alphabet, while Section 5 addresses the traditional form of model matching. A detailed example runs through the entire discussion, demonstrating notions and constructions.

2. Fundamentals

2.1. Notation

Let $\Sigma = (A, X, f)$ be an input/state asynchronous machine. Recall that the recursion function $f : X \times A \to X$ is, in general, only a partial function; it may not be defined over its entire domain. A state/input pair $(x, v) \in X \times A$ is a *valid pair* of Σ if f is defined at (x, v). A valid pair (x, v) is a *stable combination* if $f(x, v) = x$, namely, if Σ lingers at x until the input character v is changed; in such case, we often refer to x as a *stable state*. On the other hand, if $f(x, v) \neq x$, then the pair (x, v) is a *transient combination*; the machine cannot linger at x when the input is v. An asynchronous machine passes through a transient combination very quickly, ideally in zero time.

Consider the case where the machine Σ is at a stable combination (x, v'), when the input character v' changes to a different character, say, the character v. This change gives rise to a chain of transitions

$$x_1 = f(x, v),\ x_2 = f(x_1, v),\ x_3 = f(x_2, v), \ldots, \tag{3}$$

which may or may not terminate (e.g. Kohavi 1978). If this chain of transitions does not terminate, then Σ has an infinite cycle. The control of asynchronous machines with infinite cycles is considered in Venkatraman and Hammer (2006a, 2006b, 2006c); in the present paper, we restrict our attention to asynchronous machines with no infinite cycles, namely, we adopt the following.

Convention 1: Asynchronous machines considered in this paper have no infinite cycles.

Under Convention 1, the chain of transitions (3) must terminate. This means that there is an integer $i \geq 1$ such that $x_i = f(x_i, v)$. Then, (x_i, v) is a stable combination, and the state $x' := x_i$ is the *next stable state* of the pair (x, v). The transition from (x, v) to (x', v) is called a *stable transition*. If $i > 1$ in (3), then the transition from (x_j, v) to (x_{j+1}, v), $j = 0, 1, \ldots, i - 1$, where $x_0 := x$, is a *transient transition*. Thus, a stable transition may include several transient transitions along its way.

The *stable recursion function* $s : X \times A \to X$ of Σ is defined by setting $s(x, v) := x'$, where x' is the next stable state of the pair (x, v). Using s as the recursion function yields the *stable state machine* $\Sigma_{|s} := (A, X, s)$ induced by Σ. If $\Sigma_{|s} = \Sigma$, namely if $s = f$, then Σ is a *stable state machine*.

It is important to note that users of the machine Σ are aware only of its stable transitions. Transient transitions are too quick to be noticed, and too quick to have any material impact on a user. As a result, the meaningful behaviour of an asynchronous machine Σ is described by its stable recursion function s. The reason why we have to consider transient transitions in addition to stable transitions when discussing the control of an asynchronous machine Σ is that the strategy for controlling Σ involves the transformation of undesirable stable transitions of Σ into transient transitions of the closed-loop machine Σ_φ of Figure 1. The feedback function φ is designed so that only desirable stable transitions appear in Σ_φ, whereas undesirable transitions are turned into transients. The objective of the current paper is to derive necessary and sufficient conditions for the existence of such a feedback function φ.

States of the machine Σ that are not associated with any stable combination will not be noticed by a user, and, as a result, are of no practical significance. It is therefore common in the literature about asynchronous machines to ignore states that have no stable combination, and to omit such states from the state set X. We will follow this convention in our discussion.

Convention 2: Every member of the state set X of an asynchronous machine $\Sigma = (A, X, f)$ has at least one stable combination.

2.2. Fundamental mode operation

When dealing with asynchronous sequential machines, we have to be careful to avoid any operation that may result in an unpredictable response of the machine. In particular, one must avoid changing the input character while a machine is in transition. The reason for that is quite simple: if the input character of an asynchronous machine is changed while the machine undergoes a chain of transitions, asynchrony and the rapid speed at which transient transitions occur prevent an exact characterization of the machine's state at the instant at which the input character changes. Considering that the effect of an input change depends on the state of the machine at which the input change occurs, changing the input while a machine is in transition may result in an unpredictable outcome.

To avoid such potential uncertainty, a common operating policy for asynchronous machines is to prohibit changes to the input while a machine is in transition. In other words, under this operating policy, input changes are allowed only when the machine is in a stable state. Then, the state of the machine at which the input change occurs is well defined, and, as a result, so is the effect of the input change on the response of the machine (we are dealing in this paper with deterministic asynchronous machines). This brings us to the following common operating policy (e.g. Kohavi 1978).

Definition 1: An asynchronous machine is operated in *fundamental mode* if its input can change only while the machine is in a stable state.

Throughout our discussion, all asynchronous machines are operated in fundamental mode. To be specific, consider an asynchronous machine Σ with the stable recursion

function s. Assume that Σ is in a stable combination at a state x, when a string of input characters $u := u_1u_2\cdots u_k,\ k \geq 1$, is applied to it. In fundamental mode operation, the string is applied in a step-by-step manner, namely one input character at a time; the next input character is applied only after Σ has reached a stable state with the preceding input character. Thus, after the input character u_1 is applied, we wait until Σ has reached its next stable state $x_1 := s(x, u_1)$; then, the input character u_2 is applied, and we wait again until Σ has reached its next stable state $x_2 := s(x_1, u_2)$; then, the input character u_3 is applied, and so on. At each step, the next input character is applied only after confirming that Σ has reached a stable state. The outcome of this process is denoted by

$$s(x, u) = s(x, u_1u_2\cdots u_k) := s(\cdots s(s(x, u_1), u_2)\cdots u_k).$$

Denoting by A^+ the set of all non-empty strings of characters of a set A, the last equation extends the stable recursion function $s : X \times A \to X$ into a function $s : X \times A^+ \to X$.

We examine now the implications of fundamental mode operation on the use of static state feedback controllers. Recall that a static state feedback controller is represented by a function $\varphi : X \times A \to X$. This means that, for any external input character v of the closed-loop system Σ_φ of Figure 1, the function φ can assume only one value at any given state x of the controlled machine Σ. This is a substantial restriction over the flexibility afforded by a dynamic feedback controller, since the latter has its own states; its response at a pair (x, v) can vary, depending on the operational history of the closed-loop machine.

In addition to the natural restrictions that come with a static feedback controller, the selection of the feedback function φ is also restricted by the requirement of fundamental mode operation. Refer again to an asynchronous machine $\Sigma = (A, X, f)$ connected to a state feedback function $\varphi : X \times A \to X$ as depicted in Figure 1; the resulting closed-loop machine Σ_φ is described by (2). Assume that Σ is at a stable combination with a state $x_1 \in X$ while the external input character of Σ_φ is v'. Then, the input character that Σ receives from the feedback function φ is $u' := \varphi(x_1, v')$, so that (x_1, u') is a stable combination of Σ.

Consider now the case where the external input character of Σ_φ changes to v; then, the input character of Σ changes to $u := \varphi(x_1, v)$. Assume that (x_1, u) is a transient combination of Σ, and let $x_2 := f(x_1, u)$ be the next state of Σ. Assume further that (x_2, u) is still not a stable combination, and let $x_3 = f(x_2, u)$ be the succeeding state of Σ. Now, if $\varphi(x_2, v) \neq \varphi(x_1, v)$, then we have a situation where the input of Σ changes while Σ is in transition. Indeed, it takes some time for Σ to transition from the state x_2 to its next state, and it takes some time for the value of φ to change from $\varphi(x_1, v)$ to $\varphi(x_2, v)$. In an asynchronous environment, these times are unrelated. If Σ transitions from x_2 before φ has changed its value, then the next state of Σ will be x_3; on the other hand, if φ changes its value while Σ is still at the state x_2, then the next state of Σ will be $x_3' := f(x_2, \varphi(x_2, v))$. When $\varphi(x_2, v) \neq u$, it may very well happen that $x_3' \neq x_3$, thus creating a potential uncertainty. Clearly, this forms a violation of fundamental mode operation. Consequently, to preserve fundamental mode operation, we must have that $\varphi(x_2, v) = \varphi(x_1, v)$.

To fully examine the consequences of this observation, let $(x_1, u), (x_2, u), (x_3, u), \ldots, (x_i, u), i > 2$, be the chain of transitions from the pair (x_1, u) to the next stable combination (x_i, u), where $x_{j+1} = f(x_j, u), j = 1, 2, \ldots, i-1$. Then, $(x_1, u), (x_2, u), (x_3, u), \ldots, (x_{i-1}, u)$ are all transient combinations. Let v be the external input character of the closed-loop

machine Σ_φ of Figure 1. By fundamental model operation, the external input character v must be kept constant until Σ_φ has reached its next stable state (which happens very quickly; ideally, in zero time). According to the argument of the previous paragraph, fundamental mode operation also dictates that we must have $\varphi(x_1, v) = \varphi(x_2, v) = \cdots = \varphi(x_{i-1}, v) = u$. This constant value u of the feedback function φ will then take Σ to the next stable combination (x_i, u). At the state x_i, the machine Σ has reached a stable state, and the value of φ can change. In other words, we can have $\varphi(x_i, v) \neq u$; fundamental mode operation imposes no restriction on the value $\varphi(x_i, v)$. From an operational standpoint, the state x_i is reached while φ still produces the input character u for Σ; once x_i is reached, φ senses the pair (x_i, v), and, as Σ is at the stable state x_i, the value of φ can change. This argument proves the following statement which underlies our discussion in this paper.

Proposition 1: *Let $\Sigma = (A, X, f)$ be an input/state asynchronous machine and let $\varphi : X \times A \to X$ be a function. Then, the following two statements are equivalent.*

(i) *The closed-loop asynchronous machine Σ_φ operates in fundamental mode.*

(ii) *For every valid pair (x, v) of the closed-loop machine Σ_φ, the function φ satisfies $\varphi(x, v) = \varphi(f(x, \varphi(x, v)), v)$ whenever the next step $(f(x, \varphi(x, v)), \varphi(x, v))$ is a transient combination of Σ.*

Referring to Proposition 1, note that when the next step $(f(x, \varphi(x, v)), \varphi(x, v))$ of Σ forms a stable combination, there is no restriction on the value of φ at the pair $(f(x, \varphi(x, v)), v)$, namely there is no restriction on the value $\varphi(f(x, \varphi(x, v)), v)$. On the other hand, when the next step $(f(x, \varphi(x, v)), \varphi(x, v))$ is a transient combination of Σ, then φ must have the same value at the two pairs (x, v) and $(f(x, \varphi(x, v)), v)$; otherwise, the input character of Σ will change during a transient, in violation of fundamental mode operation.

The requirement of Proposition 1(ii) restricts the class of feedback functions φ that can be used for the machine Σ. This imposes a restriction on the class of models that can be matched when using static state feedback controllers. As a result, the class of models that can be matched by static state feedback is somewhat smaller than the class of models that can be matched with dynamic state feedback controllers, a class of models characterized in Murphy, Geng, and Hammer (2002, 2003). In intuitive terms, it is easy to see why static state feedback controllers are more restrictive: a dynamic controller is affected by three variables – the external input character v, the state x of Σ, and the internal state of the controller itself; a static controller, on the other hand, is affected only by two variables – the external input character v and the state x of Σ.

Example 1: Consider a simple asynchronous machine $\Sigma = (A, X, f)$ that represents a home security system and is described by the state flow diagram of Figure 2; here, $A = \{a, b, c, d\}$ and $X = \{x^1, x^2, x^3, x^4\}$. The initial state of the machine is x^1, and the input character c represents a reset of the machine by its owner. Input characters a, b, and d represent potential break-in points, such as a window, a rear door and a terrace. The alarm system has three alarm states represented by stable states at x^2, x^3 and x^4. Stable states at x^2 and x^4 represent highest alarm states and cannot be reset by the owner (to prevent tampering).

An examination of Figure 2 shows that the recursion function $f : X \times A \to X$ of Σ is given by Table 1 and the stable recursion function $s : X \times A \to X$ of Σ is given by Table 2.

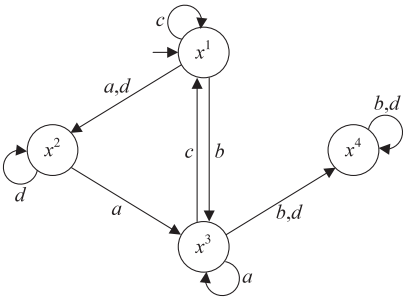

Figure 2. The state flow diagram of Σ.

Table 1. The recursion function f of Σ.

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

Table 2. The stable recursion function s of Σ.

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

For this machine, applying the input character b at the state x^1 results in the chain of transitions $(x^1, b)(x^3, b)(x^4, b)$, where the first two pairs are transient pairs and the last pair is a stable combination. In view of Proposition 1, any feedback function $\varphi : X \times A \to A$ that yields a closed-loop machine Σ_φ that operates in fundamental mode must satisfy the following requirement: If there is an external input character $v \in A$ for which $\varphi(x^1, v) = b$, then we must also have that $\varphi(x^3, v) = b$. On the other hand, the value $\varphi(x^4, v)$ can be any character of A that forms a valid pair with x^4; according to Table 1, this value can be any one of the characters b or d.

2.3. Operations on transition chains

Let $\Sigma = (A, X, f)$ be an asynchronous machine with the stable recursion function s. We say that a state x' is *stably reachable* from a state x if there is a string $u \in A^+$ such that $x' = s(x, u)$. Based on this notion, the matrix of stable transitions was used in Murphy, Geng, and Hammer (2002, 2003) to characterize all pairs of states that are stably reachable from each other. As seen there, the matrix of stable transitions is a critical tool for characterizing the ways in which dynamic state feedback controllers can modify the behaviour of the machine Σ.

An examination of Proposition 1 leads us to the conclusion that, in order to examine the capabilities of static state feedback controllers, we must take into consideration not

only the endpoints of a stable transition, but also the transient states encountered along the way. For a given external input character v, the value of the feedback function $\varphi(x, v)$ on a particular state x may be determined by an unrelated stable transition in which x is a transient state. In other words, for a given external input character v, the closed-loop machine Σ_φ can implement only stable transitions that either have no transient or stable states in common; or, if they do have states in common, the values of φ on these states must be the same. As a result, when considering static state feedback controllers, we must examine entire chains of transitions, not just stable endpoints.

To accommodate this requirement, it is convenient to introduce a function that produces the chain of transitions induced by a given state/input pair. Referring to the asynchronous machine $\Sigma = (A, X, f)$ with the stable recursion function s, let $(x, v) \in X \times A$ be a valid pair of Σ, and let $x_1 = f(x, v), x_2 = f(x_1, v), x_3 = f(x_2, v), \cdots, x_i = f(x_i, v)$ be the chain of transitions from the pair (x, v) to the next stable state $x_i = s(x, v)$, $i \geq 1$. This induces the string of pairs $(x, v)(x_1, v)(x_2, v) \cdots (x_i, v)$; here, we adopt the convention that stable combinations at the end of a transition chain are listed only once. Then, no state can appear more than once in such a string of pairs, if the machine Σ has no infinite cycles.

Now, let N be a character not in any of the sets A or X; we use N to indicate the lack of a transition. Then, using the above notation, define the function $\tau : X \times A \to (X \times A)^+ \cup N : (x, v) \mapsto \tau(x, v)$ by setting:

$$\tau(x, v) = \begin{cases} (x, v)(x_1, v) \cdots (x_{i-1}, v)(x_i, v) & \text{if } (x, v) \text{ is a valid pair,} \\ N & \text{otherwise;} \end{cases} \tag{4}$$

here, $x_i = s(x, v)$ is the next stable state of the pair (x, v), and $x_1 \neq x$ and $x_{i-1} \neq x_i$, so that no duplicates appear in the string of pairs listed in the first line of (4). Then, $\tau(x, v)$ generates the string of all state/input pairs traversed by Σ as it undergoes the stable transition from (x, v) to $s(x, v)$. We refer to τ as the *transition chain function* of the machine Σ.

Example 2: Referring to the asynchronous machine Σ of Example 1, we have seen in that example that $\tau(x^1, b) = (x^1, b)(x^3, b)(x^4, b)$.

Next, denoting by $P(X \times A)$ the set of all subsets of $X \times A$, we split the chain of pairs of (4) into a simple set of pairs using the *splitting operator* $\Delta : (X \times A)^+ \cup N \to P(X \times A)$ defined by

$$\Delta(a) := \begin{cases} \{(x, v), (x_1, v), \cdots, (x_i, v)\} & \text{if } a = (x, v)(x_1, v) \cdots (x_{i-1}, v)(x_i, v) \\ & \text{and } i \geq 1; \\ \{(x, v)\} & \text{if } a = (x, v); \\ \varnothing & \text{if } a = N. \end{cases}$$

Example 3: Continuing from Example 2, we have that $\Delta((x^1, b)(x^3, b)(x^4, b)) = \{(x^1, b), (x^3, b), (x^4, b)\}$.

It is sometimes necessary in our discussion to delete the last member of a transition chain; for this purpose, we define the operator $\Delta^- : (X \times A)^+ \cup N \to P(X \times A)$ that

truncates the last pair from the outcome of the splitting operator, as follows.

$$\Delta^{-}(a) := \begin{cases} \{(x,v),(x_1,v),\cdots,(x_{i-1},v)\} & \text{if } a = (x,v)(x_1,v)\cdots(x_{i-1},v)(x_i,v) \\ & \text{and } i > 1; \\ \{(x,v)\} & \text{if } a = (x,v)(x_1,v); \\ \varnothing & \text{if } a = N \text{ or if } a = (x,v). \end{cases}$$

Note that the third case in the definition of $\Delta^{-}(a)$ includes the case where a is itself a stable combination.

Example 4: Referring to Example 2, we have that $\Delta^{-}((x^1,b)(x^3,b)(x^4,b)) = \{(x^1,b),(x^3,b)\}$.

3. The chain reachability matrix

3.1. The one-step chain reachability matrix

In order to apply the condition of Proposition 1, we must examine the entire chain of transitions associated with every stable transition of the asynchronous machine Σ. The most convenient way to accomplish that is through a matrix that portrays all such transition chains. We generate such a matrix using the transition chain function τ of (4), as follows.

First, some notation. For a machine $\Sigma = (A, X, f)$ with the stable recursion function s, denote by $s[x \times A]$ the set of all next stable states of the state x, namely

$$s[x \times A] := \{x' \in X : x' = s(x,a) \text{ for a character } a \in A\}.$$

Definition 2: Let $\Sigma = (A, X, f)$ be an asynchronous machine with the state set of n states $X = \{x^1,\ldots,x^n\}$, the stable recursion function s, and the transition chain function τ, and let N be a character not in A nor in X. The *one-step chain reachability matrix* $\rho^{+}(\Sigma)$ is an $n \times n$ matrix whose (i,j) entry is

$$\rho_{ij}^{+}(\Sigma) = \begin{cases} \{\Delta(\tau(x^i,v)) : v \in A \text{ and } s(x^i,v) = x^j\} & \text{if } x^j \in s[x^i \times A]; \\ N & \text{otherwise;} \end{cases}$$

$i,j = 1,2,\ldots,n$.

Similarly, the *truncated one-step chain reachability matrix* $\rho^{-}(\Sigma)$ is given by

$$\rho_{ij}^{-}(\Sigma) = \begin{cases} \{\Delta^{-}(\tau(x^i,v)) : v \in A \text{ and } s(x^i,v) = x^j\} & \text{if } x^j \in s[x^i \times A]; \\ N & \text{otherwise;} \end{cases}$$

$i,j = 1,2,\ldots,n$.

As we can see, the entry $\rho_{ij}^{+}(\Sigma)$ consists of sets of pairs; each such set of pairs includes all state/input pairs encountered in a one-step stable transition from the state x^i to the state x^j. The entry $\rho_{ij}^{-}(\Sigma)$ also consists of sets of pairs; however, here each such set of pairs consists of all state/input pairs encountered in a one-step stable transition from the state x^i to the state x^j, except for the stable combination reached at the end.

Example 5: Consider again the machine Σ of Example 1. Here, we have $n = 4$, so $\rho^+(\Sigma)$ and $\rho^-(\Sigma)$ are both 4×4 matrices. To demonstrate the derivation of the matrix $\rho^+(\Sigma)$, note from Table 2 that Σ has a stable transition from the state x^1 to the state x^3, induced by the input character a, namely $s(x^1, a) = x^3$. Using Table 1, we can see that the chain of transitions behind this stable transition is $(x^1, a)(x^2, a)(x^3, a)$. Consequently, we have that $\tau(x^1, a) = (x^1, a)(x^2, a)(x^3, a)$, and it follows that

$$\rho^+_{1,3}(\Sigma) = \{\Delta\tau(x^1, a)\} = \{(x^1, a), (x^2, a), (x^3, a)\}.$$

Applying a similar process to each one-step stable transition of Σ, we obtain the matrix

$$\rho^+(\Sigma) = \begin{pmatrix} \{(x^1,c)\} & \{(x^1,d),(x^2,d)\} & \{(x^1,a),(x^2,a),(x^3,a)\} & \{(x^1,b),(x^3,b),(x^4,b)\} \\ N & \{(x^2,d)\} & \{(x^2,a),(x^3,a)\} & N \\ \{(x^3,c),(x^1,c)\} & N & \{(x^3,a)\} & \{(x^3,b),(x^4,b)\},\{(x^3,d),(x^4,d)\} \\ N & N & N & \{(x^4,b)\},\{(x^4,d)\} \end{pmatrix}.$$

To obtain the truncated one-step chain reachability matrix $\rho^-(\Sigma)$, we simply exclude the stable combination at the end of each entry of $\rho^+(\Sigma)$; this yields the matrix

$$\rho^-(\Sigma) = \begin{pmatrix} N & \{(x^1, d)\} & \{(x^1, a), (x^2, a)\} & \{(x^1, b), (x^3, b)\} \\ N & N & \{(x^2, a)\} & N \\ \{(x^3, c)\} & N & N & \{(x^3, b)\}, \{(x^3, d)\} \\ N & N & N & N \end{pmatrix}.$$

The following statement is a direct consequence of Definition 2.

Proposition 2: *Let $\Sigma = (A, X, f)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and the truncated one-step chain reachability matrix $\rho^-(\Sigma)$. Then, the following is true for any pair of integers $i \neq j \in \{1, 2, \ldots, n\}$:*

$\rho^-_{ij}(\Sigma) \neq N$ if and only if there is a one-step stable transition from the state x^i to the state x^j.

$\rho^+_{ij}(\Sigma) \neq N$ if and only if there is a one-step stable transition from the state x^i to the state x^j.

3.2. Implementation considerations

We turn now to the question of when can a collection of stable transitions be implemented by a static state feedback controller. To this end, we employ two common projections: $\Pi_X : X \times A \to X : \Pi_X(x, a) \mapsto x$ is the projection that extracts the state x from a state/input pair (x, a); similarly, $\Pi_A : X \times A \to A : \Pi_A(x, a) \mapsto a$ the projection that extracts the input character a from a state/input pair (x, a). We expand the domain of these two projections from the set $X \times A$ to the set $(X \times A) \cup N$ by setting

$$\Pi_X N = \varnothing, \Pi_A N = \varnothing,$$

where $\varnothing$ is the empty set. The following notion is critical to our discussion.

Definition 3: Let $S \subseteq (X \times A) \cup N$ be a set that may include state/input pairs and the character N. Then, S is an *implementable set* if the following is true for any members $\alpha, \alpha' \in S$: if $\Pi_A\alpha \neq \Pi_A\alpha'$, then also $\Pi_X\alpha \neq \Pi_X\alpha'$.

In other words, a set S is implementable if distinct pairs in S always have distinct states; or, equivalently, if each state is paired in S with at most one input character.

Example 6: Using the state set and the input set of Example 1, it follows that $\{(x^1, a), (x^2, a), (x^3, b)\}$ is an implementable set, while $\{(x^1, a), (x^1, b), (x^3, c)\}$ is not an implementable set.

When $S \subseteq (X \times A) \cup N$ is an implementable set, we can use the members of S to define a function $\phi : X \to A$ by setting $\phi(x) := a$ for every pair $(x, a) \in S$. A slight reflection shows that, when S is not an implementable set, such a function does not exist, since, in such case, a single member of X is assigned to two or more different characters of A. This leads us to the following simple statement, which underlies much of our discussion.

Proposition 3: *Let $S \subseteq (X \times A) \cup N$ be a non-empty set. Then, the following two statements are equivalent.*

(i) *There is a function $\phi : X \to A$ such that $(x, \phi(x)) \in S$ for all $x \in \Pi_X S$.*
(ii) *S is an implementable set.*

Example 7: Using the implementable set $\{(x^1, a), (x^2, a), (x^3, b)\}$ of Example 6, we can define a partial function $\phi : X \to A$ by setting $\phi(x^1) := a$, $\phi(x^2) := a$, $\phi(x^3) := b$. This partial function can then be extended into a function over the entire set X by setting, for example, $\phi(x^4) := d$.

Implementable sets play a critical role in our discussion, since they form the foundation for characterizing the existence of state feedback functions. At this point, we continue our discussion of implementable sets by addressing the following question: under what circumstances is the union of two implementable sets also an implementable set? An easy-to-test answer to this question is provided by the following statement. (The symbol $\#S$ denotes the cardinality of a set S and the symbol $\setminus$ is used to denote set difference.)

Proposition 4: *Let $S, S' \subseteq (X \times A) \cup N$ be implementable sets. Then, the union $S \cup S'$ is an implementable set if and only if $\#[(S \cap S') \setminus N] = \#(\Pi_X S \cap \Pi_X S')$.*

Proof: A slight reflection shows that the equality $\#[(S \cap S') \setminus N] = \#(\Pi_X S \cap \Pi_X S')$ is valid if and only if every state that appears in pairs that are common to S and S' is paired with the same input character in both S and S'. Hence, the union $S \cup S'$ includes no pairs in which the same state is paired with different input characters. This proves the assertion. □

The condition of Proposition 4 leads us to a new operation between sets of pairs, which combines sets of pairs whenever the outcome is an implementable set; otherwise, it generates the character N.

Definition 4: The *compatible union* $S \sqcup S'$ of two sets $S, S' \subseteq (X \times A) \cup N$ is given by

$$S \sqcup S' := \begin{cases} N & \text{if } S = N; \\ N & \text{if } S' = N; \\ N & \text{if } \#[(S \cap S') \setminus N] \neq \#(\Pi_X S \cap \Pi_X S'); \\ S \cup S' & \text{otherwise.} \end{cases} \tag{5}$$

Combining Definition 4 with Proposition 4, we obtain the following fact.

Proposition 5: *Let $S, S' \subseteq (X \times A) \cup N$ be implementable sets. Then, the compatible union $S \sqcup S'$ is an implementable set whenever it is not N.*

Example 8: Consider, for example, the two entries $S := \rho^+_{12}(\Sigma) = \{(x^1, d), (x^2, d)\}$ and $S' := \rho^+_{23}(\Sigma) = \{(x^2, a), (x^3, a)\}$ of Example 5. Here, $S \cap S' = \varnothing$; also, $\Pi_X S = \{x^1, x^2\}$ and $\Pi_X S' = \{x^2, x^3\}$, so we have $\Pi_X S \cap \Pi_X S' = \{x^2\}$. Consequently, $\#[(S \cap S') \setminus N] = 0 \neq \#(\Pi_X S \cap \Pi_X S') = 1$, and it follows from (5) that $S \sqcup S' = N$.

On the other hand, we have $\rho^+_{13}(\Sigma) = \{(x^1, a), (x^2, a), (x^3, a)\}$ and $\rho^+_{23}(\Sigma) = \{(x^2, a), (x^3, a)\}$, so that $\#[(\rho^+_{13}(\Sigma) \cap \rho^+_{23}(\Sigma)) \setminus N] = \#\{(x^2, a), (x^3, a)\} = 2$ and $\#[(\Pi_X \rho^+_{13}(\Sigma) \cap \Pi_X \rho^+_{23}(\Sigma))] = \#\{x^2, x^3\} = 2$; consequently, $\rho^+_{13}(\Sigma) \sqcup \rho^+_{23}(\Sigma) = \{(x^1, a), (x^2, a), (x^3, a)\}$.

Another convenient notational tool is the following, which simply turns two sets $S_1, S_2 \subseteq (X \times A) \cup N$ into a list of two members:

$$S_1 \oplus S_2 := \begin{cases} \{S_1, S_2\} & \text{if } S_1 \neq N \text{ and } S_2 \neq N; \\ S_1 & \text{if } S_2 = N; \\ S_2 & \text{if } S_1 = N. \end{cases} \tag{6}$$

Example 9: Using the sets of Example 8, we have simply that $\rho^+_{13}(\Sigma) \oplus \rho^+_{23}(\Sigma) = \big\{\{(x^1, a), (x^2, a), (x^3, a)\}, \{(x^2, a), (x^3, a)\}\big\}$.

Given a collection of sets $S_1, S_2, \ldots, S_m \subseteq (X \times A) \cup N$, where $m \geq 2$, we use the notation

$$\bigoplus_{i=1,2,\ldots,m} S_i := \begin{cases} \{\bigoplus_{i=1,2,\ldots,m-1} S_i, S_m\} & \text{if } \bigoplus_{i=1,2,\ldots,m-1} S_i \neq N \text{ and } S_m \neq N; \\ \bigoplus_{i=1,2,\ldots,m-1} S_i & \text{if } S_m = N; \\ S_m & \text{if } \bigoplus_{i=1,2,\ldots,m-1} S_i = N. \end{cases}$$

Finally, the *compatible union* of two lists $\{S_1, S_2, \ldots, S_m\}$ and $\{S'_1, S'_2, \ldots, S'_{m'}\}$ of subsets of $(X \times A) \cup N$ is defined by

$$\{S_1, S_2, \ldots, S_m\} \sqcup \{S'_1, S'_2, \ldots, S'_{m'}\} := \bigoplus_{\substack{i=1,\ldots,m, \\ j=1,\ldots,m'}} S_i \sqcup S'_j,$$

namely the list of all compatible unions of any two members. Note that every member of $\{S_1, S_2, \ldots, S_m\} \sqcup \{S'_1, S'_2, \ldots, S'_{m'}\}$ is an implementable set, unless the result of the entire operation is N.

3.3. The multi-step chain reachability matrix

Using the operations introduced in the last subsection, we can define an operation akin to matrix multiplication among matrices whose entries are families of implementable sets. Let B and C be two $n \times n$ matrices whose entries are lists of implementable sets or N. Then, the *product* BC is also an $n \times n$ matrix whose entries are lists of implementable sets or N, and it is defined as follows. For every pair of integers $i, j \in \{1, 2, \ldots, n\}$, the i, j entry of the

product BC is

$$(BC)_{ij} := \left\{\{B_{i1} \sqcup C_{1j}\} \oplus \{B_{i2} \sqcup C_{2j}\} \oplus \cdots \oplus \{B_{in} \sqcup C_{nj}\}\right\}. \tag{7}$$

For future use, we record the following fact, which is a direct consequence of Proposition 4 and Definition 4 of compatible union.

Proposition 6: *Let B and C be $n \times n$ matrices whose entries are lists of implementable sets or N. Then, the product BC is also an $n \times n$ matrix whose entries are lists of implementable sets or N.*

Using the product (7), we can define powers of the one-step chain reachability matrix by setting

$$\begin{aligned} \rho^p(\Sigma) &:= \rho^+(\Sigma) \text{ for } p = 1, \\ \rho^p(\Sigma) &:= (\rho^-(\Sigma))^{p-1}\rho^+(\Sigma) \text{ for } p \geq 2. \end{aligned} \tag{8}$$

For notational purposes, it is convenient to define the zero power of $\rho(\Sigma)$ as a matrix that includes all stable combinations of Σ, as follows. Let $X = \{x^1, \ldots, x^n\}$ be the state set of the asynchronous machine Σ. For an integer $i \in \{1, 2, \ldots, n\}$, let $\sigma_i \subseteq \{x^i\} \times A$ be the set of all stable combinations of the state x^i. Then, we define the $n \times n$ matrix $\rho^0(\Sigma)$ by setting

$$\rho^0_{ij}(\Sigma) := \begin{cases} \sigma_i & \text{if } j = i, \\ N & \text{else,} \end{cases}$$

for all $i, j \in \{1, \ldots, n\}$. This notation will be handy in our ensuing discussion.

Example 10: Referring again to the machine Σ of Example 1, it follows from Table 2 that

$$\rho^0(\Sigma) = \begin{pmatrix} \{(x^1, c)\} & N & N & N \\ N & \{(x^2, d)\} & N & N \\ N & N & \{(x^3, a)\} & N \\ N & N & N & \{(x^4, b), (x^4, d)\} \end{pmatrix}.$$

Further, referring to Example 5, we have

$$\rho^1(\Sigma) = \rho^+(\Sigma).$$

Also, according to (8), we have

$$\rho^2(\Sigma) = \rho^-(\Sigma)\rho^+(\Sigma),$$

where $\rho^-(\Sigma)$ and $\rho^+(\Sigma)$ are given in Example 5. We demonstrate the calculation of this product by calculating one of the entries, say the $(1, 2)$ entry. Using (7), we have

$$\begin{aligned} \rho^2_{12}(\Sigma) &= \left(\rho^-(\Sigma)\rho^+(\Sigma)\right)_{12} \\ &= \left\{\{\rho^-_{11}(\Sigma) \sqcup \rho^+_{12}(\Sigma)\} \oplus \{\rho^-_{12}(\Sigma) \sqcup \rho^+_{22}(\Sigma)\} \oplus \{\rho^-_{13}(\Sigma) \sqcup \rho^+_{32}(\Sigma)\}\right. \\ &\quad \left. \oplus \{\rho^-_{14}(\Sigma) \sqcup \rho^+_{42}(\Sigma)\}\right\} \\ &= \left\{N \oplus \{(x^1, d), (x^2, d)\} \oplus N \oplus N\right\} \\ &= \{(x^1, d), (x^2, d)\}. \end{aligned}$$

Calculating the remaining entries in a similar fashion, we obtain

$$\rho^2(\Sigma) = \rho^-(\Sigma)\rho^+(\Sigma) =$$

$$\begin{pmatrix} N & \{(x^1,d),(x^2,d)\} & \rho^2_{1,3}(\Sigma) & \rho^2_{1,4}(\Sigma) \\ \{(x^2,a),(x^3,c),(x^1,c)\} & N & \{(x^2,a),(x^3,a)\} & \rho^2_{2,4}(\Sigma) \\ \{(x^3,c),(x^1,c)\} & \{(x^3,c),(x^1,d),(x^2,d)\} & N & \rho^2_{3,4}(\Sigma) \\ N & N & N & N \end{pmatrix},$$

where

$$\begin{aligned}
\rho^2_{1,3}(\Sigma) &= \left\{\{(x^1,d),(x^2,a),(x^3,a)\},\{(x^1,a),(x^2,a),(x^3,a)\}\right\};\\
\rho^2_{1,4}(\Sigma) &= \left\{\{(x^1,a),(x^2,a),(x^3,b),(x^4,b)\},\{(x^1,a),(x^2,a),(x^3,d),(x^4,d)\},\right.\\
&\qquad \left.\{(x^1,b),(x^3,b),(x^4,b)\},\{(x^1,b),(x^3,b),(x^4,d)\}\right\};\\
\rho^2_{2,4}(\Sigma) &= \left\{\{(x^2,a),(x^3,b),(x^4,b)\},\{(x^2,a),(x^3,d),(x^4,d)\}\right\};\\
\rho^2_{3,4}(\Sigma) &= \left\{\{(x^3,b),(x^4,b)\},\{(x^3,b),(x^4,d)\},\{(x^3,d),(x^4,b)\},\{(x^3,d),(x^4,d)\}\right\}.
\end{aligned}$$

Continuing in this pattern, we get

$$\begin{aligned}
\rho^3(\Sigma) &= \left(\rho^-(\Sigma)\right)^2\rho^+(\Sigma) = \rho^-(\Sigma)\rho^2(\Sigma)\\
&= \begin{pmatrix} N & N & \{(x^1,d),(x^2,a),(x^3,a)\} & \rho^3_{1,4}(\Sigma) \\ \{(x^2,a),(x^3,c),(x^1,c)\} & N & N & \rho^3_{2,4}(\Sigma) \\ N & \{(x^3,c),(x^1,d),(x^2,d)\} & N & N \\ N & N & N & N \end{pmatrix},
\end{aligned}$$

where

$$\begin{aligned}
\rho^3_{1,4}(\Sigma) &= \left\{\{(x^1,d),(x^2,a),(x^3,b),(x^4,b)\},\{(x^1,d),(x^2,a),(x^3,d),(x^4,d)\},\right.\\
&\qquad \{(x^1,a),(x^2,a),(x^3,b),(x^4,b)\},\{(x^1,a),(x^2,a),(x^3,b),(x^4,d)\},\\
&\qquad \left.\{(x^1,a),(x^2,a),(x^3,d),(x^4,b)\},\{(x^1,a),(x^2,a),(x^3,d),(x^4,d)\}\right\};\\
\rho^3_{2,4}(\Sigma) &= \left\{\{(x^2,a),(x^3,b),(x^4,b)\},\{(x^2,a),(x^3,b),(x^4,d)\},\{(x^2,a),(x^3,d),(x^4,b)\},\right.\\
&\qquad \left.\{(x^2,a),(x^3,d),(x^4,d)\}\right\}.
\end{aligned}$$

We will see later that it is not necessary in this case to computer higher powers of $\rho(\Sigma)$.

Next, we define an operation akin to matrix addition. As before, let B and C be two $n \times n$ matrices whose entries are lists of sets or N. Then, the sum $B \oplus C$ of B and C is also an $n \times n$ matrix whose entries are lists of sets or N, where the i,j entry is

$$(B \oplus C)_{ij} := B_{ij} \oplus C_{ij},$$

$i,j \in \{1,2,\ldots,n\}$. In view of (6), the following is true.

Proposition 7: *Let B and C be $n \times n$ matrices whose entries are lists of implementable sets or N. Then, the sum $B \oplus C$ is also an $n \times n$ matrix whose entries are lists of implementable sets or N.*

Finally, we define the following

Definition 5: The *p-multistep chain reachability matrix* $\rho^{(p)}(\Sigma)$ of an asynchronous machine Σ is obtained by summing the powers $\rho^0(\Sigma), \rho(\Sigma), \rho^2(\Sigma), \ldots, \rho^p(\Sigma)$ of the chain reachability matrix of Σ up to a step $p \geq 0$:

$$\rho^{(p)}(\Sigma) := \bigoplus_{i=0,1,\ldots,p} \rho^i(\Sigma). \tag{9}$$

The p-multistep reachability matrix plays an important role in our discussion.

Example 11: Using the data of Example 10, we obtain (after removing duplicate members)

$$\begin{aligned}\rho^{(3)}(\Sigma) &= \rho^0(\Sigma) \oplus \rho^1(\Sigma) \oplus \rho^2(\Sigma) \oplus \rho^3(\Sigma) \\ &= \begin{pmatrix} \{(x^1,c)\} & \{(x^1,d),(x^2,d)\} & \rho_{1,3}(\Sigma) & \rho_{1,4}(\Sigma) \\ \{(x^2,a),(x^3,c),(x^1,c)\} & \{(x^2,d)\} & \{(x^2,a),(x^3,a)\} & \rho_{2,4}(\Sigma) \\ \{(x^3,c),(x^1,c)\} & \{(x^3,c),(x^1,d),(x^2,d)\} & \{(x^3,a)\} & \rho_{3,4}(\Sigma) \\ N & N & N & \{(x^4,b),(x^4,d)\} \end{pmatrix}\end{aligned}$$

where

$$\begin{aligned}\rho_{1,3}(\Sigma) &= \big\{\{(x^1,d),(x^2,a),(x^3,a)\},\{(x^1,a),(x^2,a),(x^3,a)\}\big\}; \\ \rho_{1,4}(\Sigma) &= \big\{\{(x^1,b),(x^3,b),(x^4,b)\},\{(x^1,b),(x^3,b),(x^4,d)\}, \\ &\quad \{(x^1,d),(x^2,a),(x^3,b),(x^4,b)\},\{(x^1,d),(x^2,a),(x^3,d),(x^4,d)\}, \\ &\quad \{(x^1,a),(x^2,a),(x^3,b),(x^4,b)\},\{(x^1,a),(x^2,a),(x^3,b),(x^4,d)\}, \\ &\quad \{(x^1,a),(x^2,a),(x^3,d),(x^4,b)\},\{(x^1,a),(x^2,a),(x^3,d),(x^4,d)\}\big\}; \\ \rho_{2,4}(\Sigma) &= \big\{\{(x^2,a),(x^3,b),(x^4,b)\},\{(x^2,a),(x^3,d),(x^4,d)\}, \\ &\quad \{(x^2,a),(x^3,b),(x^4,d)\}\big\}; \\ \rho_{3,4}(\Sigma) &= \big\{\{(x^3,b),(x^4,b)\},\{(x^3,b),(x^4,d)\},\{(x^3,d),(x^4,b)\},\{(x^3,d),(x^4,d)\}\big\}.\end{aligned} \tag{10}$$

The significance of the p-multistep reachability matrix originates from the fact that every entry that is not N represents a transition that can be implemented by static state feedback, as described in the following statement. This statement forms the foundation of our discussion in this paper.

Theorem 1: *Let $\Sigma = (A, X, f)$ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$. Then, the following two statements are equivalent.*

(i) *There is an integer $p \geq 1$ and a pair of integers $i, j \in \{1, 2, \ldots, n\}$ for which $\rho_{ij}^{(p)}(\Sigma) \neq N$.*

(ii) *There is a state feedback function $\phi : X \to A$ for which the autonomous closed-loop machine Σ_ϕ has a stable transition from x^i to x^j in fundamental mode operation.*

Proof: We start by showing that (i) implies (ii). Assuming that (i) is valid, let $p \geq 1$ be an integer, and let $i, j \in \{1, 2, \ldots, n\}$ be a pair of integers for which $\rho_{ij}^{(p)}(\Sigma) \neq N$. Then, according to (5) and (7), the entry $\rho_{ij}^{(p)}(\Sigma)$ includes a non-empty list of implementable sets; let $S \subseteq X \times A$ be one of these implementable sets. By Proposition 3, we can then

define a partial function $\phi : X \to A$ that satisfies

$$S = \{(x, \phi(x)) : x \in \Pi_X S\}.$$

If $p = 0$, then, by the definition of $\rho^0(\Sigma)$, the pair $(x, \phi(x))$ is a stable combination for all $x \in \Pi_X S$, and the feedback function ϕ simply keeps Σ at the corresponding stable combination. If $p = 1$, then the feedback function ϕ induces a one-step stable transition from the state x^i to the state x^j.

Further, to consider the case where $p > 1$, recall that each member of an entry of the truncated one-step chain reachability matrix $\rho^-(\Sigma)$ originates from a transition chain that takes Σ from one stable combination to the next stable combination, where the stable combination at the end of the transition chain has been omitted. Also, each member of an entry of the one-step chain reachability matrix $\rho^+(\Sigma)$ originates from a transition chain that takes Σ from one stable combination to the next stable combination, but here the last stable combination is included. Considering the definition of matrix multiplication (see (7)), we conclude that the set S originates from p consecutive transition chains of Σ, where each transition chain represents a one-step stable transition of Σ; except for the last step, Σ proceeds to the next step while in the last stable state of the previous step. Let us denote the p transition chains from which S originates as follows:

$$\begin{array}{c} (x_{1,1}, v_1)(x_{1,2}, v_1) \cdots (x_{1,k_1}, v_1); \\ (x_{2,1}, v_2)(x_{2,2}, v_2) \cdots (x_{2,k_2}, v_2); \\ \cdots \\ (x_{p,1}, v_p)(x_{p,2}, v_p) \cdots (x_{p,k_p}, v_p). \end{array} \tag{11}$$

Here, each row represents a one-step stable transition; for each integer $\ell = 1, 2, \ldots, p$, the pair $(x_{\ell,k_\ell}, v_\ell)$ is the next stable combination of the pair $(x_{\ell,1}, v_\ell)$. As these form consecutive stable transitions, we have that

$$x_{\ell,k_\ell} = x_{\ell+1,1} \quad \text{for all } \ell = 1, 2, \ldots, p-1. \tag{12}$$

Considering that the matrix $\rho^-(\Sigma)$ omits the last stable pair in each transition chain, it follows that the pairs that actually appear in S are the following:

$$\begin{array}{c} (x_{1,1}, v_1), (x_{1,2}, v_1), \cdots, (x_{1,k_1-1}, v_1), \\ (x_{2,1}, v_2), (x_{2,2}, v_2), \cdots, (x_{2,k_2-1}, v_2), \\ \cdots \\ (x_{p-1,1}, v_{p-1}), (x_{p-1,2}, v_{p-1}), \cdots, (x_{p-1,k_2-1}, v_{p-1}), \\ (x_{p,1}, v_p), (x_{p,2}, v_p), \cdots, (x_{p,k_p}, v_p). \end{array} \tag{13}$$

Note that the last chain does include the stable combination at the end, since the last term originates from $\rho^+(\Sigma)$. Using the data of (13), the function ϕ is given by

$$\begin{aligned} \phi(x_{a,b}) &= v_a, b = 1, 2, \ldots, (k_a - 1), a = 1, 2, \ldots, p, \\ \phi(x_{p,k_p}) &= v_p. \end{aligned}$$

Here, the integer a is fixed along each chain of transitions from one stable combination to the next, while b indicates the transient states encountered along the way. At the end of the last stable transition, namely when $a = p$ and $b = k_p$, the relation $\phi(x_{p,k_p}) = v_p$ makes the closed-loop machine Σ_ϕ stay indefinitely at the stable combination (x_{p,k_p}, v_p) of Σ, which forms the end of the entire process.

When $p = 1$, then ϕ has only one value, namely the character v_1. If $p > 1$, then, for every integer $a = 1, 2, \ldots, p-1$, the function ϕ changes its value from v_a to v_{a+1} at the state x_{a,k_a}; this change starts the chain of transitions of the next stable step (see (12)). As this change in the value of ϕ occurs at a stable state of Σ, fundamental mode operation is preserved (see Proposition 1). At the last state x_p, the feedback function ϕ maintains the value v_p, thus keeping Σ in the stable combination (x_{p,k_p}, v_p). As a result of these values of ϕ, the closed-loop system Σ_ϕ moves through the entire string of consecutive transition chains given by (11), going from the state $x_{1,1}$ to the state x_{p,k_p} and resting at the last stable combination (x_{p,k_p}, v_p). Hence, the feedback function ϕ induces the required stable transition without violating fundamental mode operation. This proves that (i) implies (ii).

Conversely, assume that (ii) is valid. Then, there must be a string of stable transitions that takes the machine Σ from the state x^i to the state x^j. Considering that this string of stable transitions is implemented by a static feedback function ϕ, it follows by Proposition 3 that the state/input pairs encountered along this string of stable transitions must form an implementable set. In other words, there must be an integer $p \geq 1$ and a string of stable transitions as described by (11). In view of the discussion surrounding (13), this implies that $\rho_{ij}^{(p)}(\Sigma) \neq N$. Consequently, (i) is valid and our proof concludes. □

The following statement shows that it is sufficient to consider p-multistep chain reachability matrices up to step $p = n - 1$. This result is analogous to a statement that appears in a different context in Murphy, Geng, and Hammer (2002, 2003).

Theorem 2: *Let $\Sigma = (A, X, f)$ be an input/state asynchronous machine with n states. Then, the following two statements are equivalent for all integers $i, j \in \{1, 2, \ldots, n\}$.*

(i) $\rho_{ij}^{(n-1)}(\Sigma) \neq N$

(ii) $\rho_{ij}^{(p)}(\Sigma) \neq N$ *for an integer* $p \geq 0$.

Proof: Let $X = \{x^1, x^2, \ldots, x^n\}$ be the state set of Σ. Consider a pair of integers $i, j \in \{1, 2, \ldots, n\}$, and let $p \geq n$ be an integer. Clearly, if (i) is valid, then it follows directly from the definition of $\rho_{ij}^{(p)}(\Sigma)$ (see Definition 5) that (ii) is valid for all integers $p \geq n - 1$. Consequently, (i) implies (ii).

Conversely, assume that (ii) is valid for an integer $p \geq 0$. If $p \leq n - 1$, then (i) is also valid by the definition of $\rho_{ij}^{(n-1)}(\Sigma)$ (see Definition 5). Therefore, it only remains to consider the case where $p \geq n$. To this end, let S be a member of $\rho_{ij}^{(p)}(\Sigma)$, and let (11) be the string of transition chains that induces S. A slight reflection shows that a transition of Σ that consists of p stable steps encounters $p + 1$ stable states along the way, including the first state $x_{1,1}$ and the last state x_{p,k_p} (recall that stable transitions always start from a stable state and end at a stable state). Specifically, in (11), the transition encompasses the list of stable states

$$\{x_{1,1}, x_{2,1}, \ldots, x_{p,1}, x_{p,k_p}\}, \tag{14}$$

which includes $p+1$ states. To make it clear that there are $p+1$ states in this list, denote

$$x_{p+1,1} := x_{p,k_p},$$
$$v_{p+1} := v_p.$$

With this notation, the list (14) becomes

$$\{x_{1,1}, x_{2,1}, \ldots, x_{p,1}, x_{p+1,1}\}. \tag{15}$$

Now, for $p \geq n$, the list (15) includes $p+1 \geq n+1$ states. As there are only n different states in the state set X of Σ, there must be a repeating state in the list (15). In other words, there are two integers $a < b$, where $a, b \in \{1, 2, \ldots, p+1\}$, such that $x_{a,1} = x_{b,1}$. Considering that all the states listed in (14) originate from the same implementable set, it follows by Definition 3 of an implementable set that the equality $x_{a,1} = x_{b,1}$ implies that $v_a = v_b$ in (11). Let us cut out the chains of transitions from $x_{a,1}$ to the stable state encountered just before $x_{b,1}$, to obtain the string of transition chains

$$\begin{gathered}(x_{1,1}, v_1)(x_{1,2}, v_1)\cdots \\ \cdots \\ (x_{a-1,1}, v_2)(x_{a-1,2}, v_2)\cdots \\ (x_{b,1}, v_b)(x_{b,2}, v_b)\cdots \\ \cdots \\ \cdots (x_{p,1}, v_p)(x_{p,2}, v_p)\cdots(x_{p,k_p}, v_p).\end{gathered}$$

Then, bearing in mind that according to our notation $(x_{p,k_p}, v_p) = (x_{p+1,1}, v_{p+1})$, this string of transition chains gives rise to a set of pairs that is a member of the entry $\rho_{ij}^{(p-(b-a))}(\Sigma)$.

Finally, if still $(p-(b-a)) \geq n$, the same process can be repeated to further shorten the string. Continuing in this manner, the string can be shortened until it yields a member of $\rho_{ij}^{(n-1)}(\Sigma)$. This concludes our proof. □

Based on Theorem 2, we define the following notion.

Definition 6: The *chain reachability matrix* of an asynchronous machine Σ with n states is the $n \times n$ matrix $\rho(\Sigma) := \rho^{(n-1)}(\Sigma)$.

Example 12: The chain reachability matrix of the asynchronous machine Σ of Example 1 is given by $\rho(\Sigma) = \rho^{(3)}(\Sigma)$, where $\rho^{(3)}(\Sigma)$ is given in Example 11.

4. Model matching through table lookup

4.1. The chain skeleton matrix

The chain reachability matrix $\rho(\Sigma)$ allows us to derive a simple solution to the problem of model matching with state feedback, but in a somewhat non-traditional way; the traditional model matching approach is considered in the next section. The approach considered in the present section yields a simpler condition for the existence of a feedback function that achieves model matching, at the cost of a somewhat more elaborate implementation.

Recall that model matching by static state feedback refers to the problem of finding a feedback function φ which, when connected around a given asynchronous machine Σ,

yields a closed-loop machine Σ_φ that matches a specified stable state machine Σ'. In the approach to model matching considered in the present section, the closed-loop machine Σ_φ does not necessarily have the same input alphabet as the model Σ' it is required to match. Instead, in order to match the response of the model Σ', the closed-loop machine Σ_φ must be operated in the following manner: for each input character v of Σ', there is a corresponding input character $a(v)$ of Σ_φ such that the response of Σ_φ to $a(v)$ is the same as the response of Σ' to v.

In this approach, operating the closed-loop machine Σ_φ so as to match the model Σ' involves the preliminary step of finding the character $a(v)$ that corresponds to the model's input character v. The collection of all corresponding pairs $(v, a(v))$ of input characters can be listed in a table. This table then serves as a 'lookup table', to be used in the process of controlling Σ_φ so as to match the model Σ'. When the two machines Σ' and Σ_φ are at the same stable state, the response of Σ' to v is obtained from Σ_φ by applying the input character $a(v)$.

We start our discussion by defining the following matrix.

Definition 7: Let Σ be an asynchronous machine with the state set $X = \{x^1, \ldots, x^n\}$ and with the chain reachability matrix $\rho(\Sigma)$. The *chain skeleton matrix* $\kappa(\Sigma)$ of Σ is an $n \times n$ matrix of zeros and ones, with the entries

$$\kappa_{ij}(\Sigma) := \begin{cases} 1 & \text{if } \rho_{ij}(\Sigma) \neq N, \\ 0 & \text{if } \rho_{ij}(\Sigma) = N, \end{cases}$$

$i, j = 1, 2, \ldots, n$.

Example 13: In Example 12, we calculated the chain reachability matrix $\rho(\Sigma)$ of the machine Σ of Example 1. Applying Definition 7 to $\rho(\Sigma)$, we obtain the chain skeleton matrix

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

of the machine Σ.

At this point, it is important to compare the notions introduced here to notions used in Murphy, Geng, and Hammer (2002, 2003), where dynamic state feedback controllers are investigated. First, recall the one-step matrix of stable transitions $R^1(\Sigma)$, which is defined as follows. Let c_{ij} be the set of all input characters that take Σ from a stable combination with the state x^i to a stable combination with the state x^j; using the stable recursion function s of Σ, we have

$$c_{ij} := \{v \in A : s(x^i, v) = x^j\}.$$

Then, letting N be a character that is not in A nor in X, the i, j entry of the $n \times n$ matrix $R^1(\Sigma)$ is given by

$$R^1_{ij}(\Sigma) = \begin{cases} c_{ij} & \text{if } c_{ij} \neq \varnothing, \\ N & c_{ij} = \varnothing, \end{cases} \tag{16}$$

$i, j = 1, 2, \ldots, n$. Based on this, the one-step skeleton matrix $K^1(\Sigma)$ introduced in Murphy, Geng, and Hammer (2002, 2003) is given by

$$K^1_{ij}(\Sigma) = \begin{cases} 1 & \text{if } R^1_{ij}(\Sigma) \neq N, \\ 0 & \text{otherwise,} \end{cases}$$

$i, j = 1, 2, \ldots, n$. Equivalently, we have

$$K^1_{ij}(\Sigma) = \begin{cases} 1 & \text{if } x^j = s(x^i, v) \text{ for some } v \in A, \\ 0 & \text{otherwise,} \end{cases}$$

$i, j = 1, 2, \ldots, n$.

Now, let $\Sigma' = (A, X, f')$ be an asynchronous machine model that must be matched by applying static state feedback to the asynchronous machine Σ. Let $K^1(\Sigma')$ be the one-step skeleton matrix of Σ', let $\kappa(\Sigma)$ be the chain skeleton matrix of Σ, and suppose that $\kappa(\Sigma) < K^1(\Sigma')$, where the inequality is taken entry by entry. Considering that these matrices include only zeros and ones, this implies that there is a pair of integers $i, j \in \{1, 2, \ldots, n\}$ for which $K^1_{ij}(\Sigma') = 1$ while $\kappa_{ij}(\Sigma) = 0$. The fact that $K^1_{ij}(\Sigma') = 1$ means that the model Σ' has a stable transition from the state x^i to the state x^j. To check whether such a transition can be implemented in Σ with a static state feedback controller, note that, by Definition 7, the equality $\kappa_{ij}(\Sigma) = 0$ means that $\rho_{ij}(\Sigma) = N$. In view of Definition 6 and Theorem 2, we conclude from this that $\rho^{(p)}_{ij}(\Sigma) = N$ for all integers $p \geq 1$. By Theorem 1, this implies that there is no state feedback function that can guide Σ through a stable transition from x^i to x^j in fundamental mode operation. As such a transition is present in the model Σ', we conclude that, when $\kappa(\Sigma) \not\geq K^1(\Sigma')$, model matching by static state feedback is not possible. However, in the opposite case, namely when $\kappa(\Sigma) \geq K^1(\Sigma')$, model matching can be achieved through the following process.

4.2. The process of table model matching

Let us now examine the case where $\kappa(\Sigma) \geq K^1(\Sigma')$. Consider a pair of integers $i, j \in \{1, \ldots, n\}$ for which $K^1_{ij}(\Sigma') = 1$. Then, we must also have that $\kappa_{ij}(\Sigma) = 1$; in view of Definition 7, this entails that $\rho_{ij}(\Sigma) \neq N$. Consequently, we can select a member $S_{ij} \in \rho_{ij}(\Sigma)$ which is not N. By Theorem 1, there is then a feedback function $\phi_{ij} : X \to A$ for which the closed-loop machine $\Sigma_{\phi_{ij}}$ has a stable transition from the state x^i to the state x^j in fundamental mode operation.

Now, let J be the set of all pairs of integers $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, n\}$ for which $K^1_{ij}(\Sigma') = 1$. Then, following the process of the previous paragraph, we can construct for each pair $(i, j) \in J$ a feedback function $\phi_{ij} : X \to A$ for which the closed-loop machine $\Sigma_{\phi_{ij}}$ has a stable transition from the state x^i to the state x^j in fundamental mode operation.

Next, let $\mathcal{A}$ be an alphabet with exactly $\#J$ characters. Assign to each pair $(i, j) \in J$ a unique character $\alpha_{ij} \in \mathcal{A}$. This induces a set isomorphism $\alpha : \mathcal{A} \cong J : \alpha_{ij} \mapsto (i, j)$.

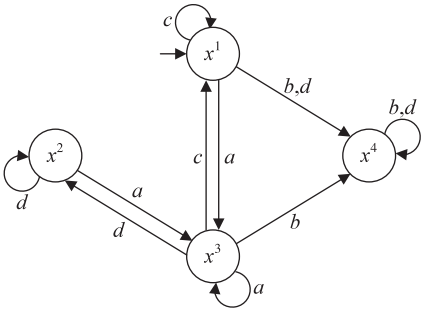

Figure 3. State flow diagram of Σ_1'.

Using the functions $\{\phi_{ij}\}$ constructed in the last two paragraphs, we can define a partial function $\varphi_t : X \times \mathcal{A} \to A$, where A is the input set of Σ, by setting

$$\varphi_t(x^i, u) := \begin{cases} \phi_{ij}(x^i) & \text{if } (i,j) \in J \text{ and } u = \alpha_{ij}; \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{17}$$

We can now match the model Σ' using the closed-loop machine Σ_{φ_t} as follows. Build a partial function $T : X \times A \to \mathcal{A}$ by setting

$$T(x^i, v) := \begin{cases} \alpha_{ij} & \text{if } s'(x^i, v) = x^j, \\ \text{undefined} & \text{otherwise,} \end{cases} \tag{18}$$

As any function over a finite set, the function T can be represented by a table.

Now, operate the closed-loop machine Σ_{φ_t} in the following manner: when Σ_{φ_t} and Σ' are both at the state x^i and the external input character v is received, apply the input character $\alpha_{ij} = T(x^i, v)$ to the closed-loop machine Σ_{φ_t}. This will then take Σ_{φ_t} to the state x^j, the same state which Σ' reaches when the input character v is applied to it. In this way, we obtain a solution to the model matching problem, although this solution is somewhat indirect, as it requires the use of the table T to translate input characters of the alphabet A to input characters of the alphabet $\mathcal{A}$. We refer to this method of model matching by the term *table model matching*. Our discussion proves the following.

Theorem 3: *Let Σ be an asynchronous machine with the chain skeleton matrix $\kappa(\Sigma)$, and let Σ' be a model with the one-step skeleton matrix $K^1(\Sigma')$. Then, the following two statements are equivalent.*

(i) *Σ can match the model Σ' through table model matching.*
(ii) $\kappa(\Sigma) \geq K^1(\Sigma')$.

Example 14: Assume that the security system represented by the machine Σ of Example 1 must be adapted to a different set-up. The newly required response is given by the asynchronous machine Σ_1' described in the state flow diagram of Figure 3. We show now that this adaptation can be achieved by table model matching. Note that the response of Σ_1' to the input characters a, b, and c is the same as that of Σ, but Σ_1' exhibits a different response to the input character d.

An examination of Figure 3 shows that Σ_1' has the stable recursion function s_1' described by Table 3.

Table 3. The stable recursion function s_1'.

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

From the stable recursion function s_1' described by Table 3, we obtain the one-step matrix of stable transitions of Σ_1':

$$R^1(\Sigma_1') = \begin{pmatrix} \{c\} & N & \{a\} & \{b,d\} \\ N & \{d\} & \{a\} & N \\ \{c\} & \{d\} & \{a\} & \{b\} \\ N & N & N & \{b,d\} \end{pmatrix}. \tag{19}$$

Based on $R^1(\Sigma_1')$, the one-step skeleton matrix of Σ_1' is

$$K^1(\Sigma_1') = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Referring to the chain skeleton matrix $\kappa(\Sigma)$ derived in Example 13, we can see right away that $\kappa(\Sigma) \geq K^1(\Sigma_1')$. Consequently, by Theorem 3, there is a feedback function φ_t that controls the machine Σ so as to match the model Σ_1' by table model matching.

To derive the feedback function φ_t, we can restrict our attention to entries of the matrix $K^1(\Sigma)$ that have the value of 1, namely to the entries

$$J := \{(1,1),(1,3),(1,4),(2,2),(2,3),(3,1),(3,2),(3,3),(3,4),(4,4)\}.$$

Retracing the procedure described in Subsection 4.2, we build an appropriate feedback functions $\phi_{ij} : X \to A$ for each pair $(i,j) \in J$. The feedback function ϕ_{ij} is built so as to induce an autonomous transition of the closed-loop machine $\Sigma_{\phi_{ij}}$ from the state x^i of Σ to the state x^j of Σ. Specifically, the function ϕ_{ij} is obtained from the state/input pairs included in a member of the corresponding entry $\rho_{ij}(\Sigma)$ of the chain reachability matrix $\rho(\Sigma)$ that was derived in Example 12. To this end, consider the pair $(1,1)$. As we have that $\rho_{11}(\Sigma) = \{(x^1, c)\}$, the (partial) function $\phi_{11} : X \to A$ is defined only at one state by

$$\phi_{11}(x^1) = c. \tag{20}$$

Next, considering the pair $(1,3)$ of J, the entry $\rho_{13}(\Sigma)$ includes many members, and any one of these members is adequate for our use; let us use the member $\{(x^1,d),(x^2,a),(x^3,a)\}$, for example. From these pairs, we obtain the (partial) function $\phi_{13} : X \to A$ given by

$$\phi_{13}(x^1) = d,\ \phi_{13}(x^2) = a,\ \phi_{13}(x^3) = a. \tag{21}$$

Table 4. The set isomorphism α.

$\mu_i \in \mathcal{A}$	$\alpha(\mu_i) \in J$
μ_1	(1, 1)
μ_2	(1, 3)
μ_3	(1, 4)
μ_4	(2, 2)
μ_5	(2, 3)
μ_6	(3, 1)
μ_7	(3, 2)
μ_8	(3, 3)
μ_9	(3, 4)
μ_{10}	(4, 4)

Table 5. The function T.

$(x, v) \in X \times A$	$T(x, v) \in \mathcal{A}$	$\alpha(\mu_i) \in J$
(x^1, c)	μ_1	(1,1)
(x^1, a)	μ_2	(1, 3)
$(x^1, b), (x^1, d)$	μ_3	(1, 4)
(x^2, d)	μ_4	(2, 2)
(x^2, a)	μ_5	(2, 3)
(x^3, c)	μ_6	(3, 1)
(x^3, d)	μ_7	(3, 2)
(x^3, a)	μ_8	(3, 3)
(x^3, b)	μ_9	(3, 4)
$(x^4, b), (x^4, d)$	μ_{10}	(4, 4)

Further, turning to the pair $(1, 4) \in J$ and using the member $\{(x^1, b), (x^3, b), (x^4, b)\}$ of $\rho_{14}(\Sigma)$, we obtain the (partial) function $\phi_{14} : X \to A$ given by

$$\phi_{14}(x^1) = b,\ \phi_{14}(x^3) = b,\ \phi_{14}(x^4) = b.$$

Continuing in this manner, we obtain all functions $\phi_{ij} : X \to A$, $(i, j) \in J$.

Now, as J has 10 elements, it follows from Subsection 4.2 that the alphabet $\mathcal{A}$ must have 10 characters as well; for example, we can use the alphabet

$$\mathcal{A} = \{\mu_1, \mu_2, \ldots, \mu_{10}\}.$$

With this alphabet, we build a set isomorphism $\alpha : \mathcal{A} \cong J$ described by Table 4. Using the one-step transition matrix $R^1(\Sigma_1')$ of the model, we can build now the function T of (18); the result is given in Table 5. Then, we can define the (partial) function $\varphi_t : X \times \mathcal{A} \to A$ by setting

$$\varphi_t(x_i, \mu_\ell) := \phi_{\alpha(\mu_\ell)}(x_i)$$

for all pairs (x_i, μ_ℓ) for which the expression $\phi_{\alpha(\mu_\ell)}(x_i)$ is meaningful.

Now, consider the closed-loop machine Σ_{φ_t} in conjunction with Table 5. Let us examine first the case when Σ_1' and Σ_{φ_t} are both in a stable combination at the state x^1, when the input character c is applied to the model Σ_1'. Then, according to Table 5, the input character μ_1 must be applied to the closed-loop machine Σ_{φ_t}. By Table 4, this results in the feedback function ϕ_{11} being applied to Σ; in view of (20), this causes the input character c to be

applied to Σ. As a result, Σ stays in a stable combination at the state x^1, simulating the response of the model Σ_1'.

To examine another case, assume that the model Σ_1' is in a stable combination at the state x^1, when the input character a is applied to it. According to Table 5, this requires us to apply the character μ_2 to Σ_{φ_t}, which results in applying to Σ the feedback function ϕ_{13}. In view of (21) and the transition table of Σ given in Table 2, this drives Σ_{φ_t} to a stable combination at the state x^3, again imitating the model Σ_1'. Continuing in this manner, we can easily verify that Σ_{φ_t} does indeed simulate the model Σ_1' by table model matching.

We turn next to traditional model matching, which requires additional considerations for the following reason. The use of the alphabet $\mathcal{A}$ makes it possible to tolerate inconsistencies among the functions $\{\phi_{ij}\}$ in the sense that two such functions may take different values on a particular state/input pair; this is thanks to the fact that each function ϕ_{ij} is associated with a different character of the alphabet $\mathcal{A}$ in the overall feedback function φ_t. However, in traditional model matching, different values of the feedback function on the same state/input pair form an inconsistency in the definition of the feedback function, and therefore cannot be tolerated. We discuss this point in the next section.

5. Traditional model matching

5.1. Necessary and sufficient conditions for traditional model matching by static state feedback

Let $\Sigma = (A, X, f)$ be an asynchronous machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, and let $\Sigma' = (A, X, s')$ be a stable state machine serving as a model. Here, Σ' has the same state set X as Σ. Considering that Σ' is a stable state machine, its stable recursion function is the same as its recursion function, namely it is s'. Our objective is to find a state feedback function $\varphi : X \times A \to A$ for which $\Sigma_\varphi = \Sigma'$, where the equality refers to the stable state machine induced by Σ_φ.

Now, let s be the stable recursion function of the asynchronous machine Σ we need to control. To examine the conditions under which Σ can match the model Σ' via static state feedback, consider the case where Σ' has two distinct one-step stable transitions with the same input character ν. Specifically, assume that the input character ν induces in Σ' a stable transition from the state x^i to the state x^j, and, in addition, it also induces in Σ' a stable transition from the state $x^{i'}$ to the state $x^{j'}$. Recalling from (16) the one-step matrix of stable transitions of Σ', this means that $\nu \in R^1_{ij}(\Sigma') \cap R^1_{i'j'}(\Sigma')$.

Now, assume that there is a static state feedback function $\varphi : X \times A \to A$ for which the closed-loop machine Σ_φ emulates the stable transitions of the model Σ', namely that $\Sigma_\varphi = \Sigma'$. Let $\rho(\Sigma)$ be the chain reachability matrix of Σ. Then, in the closed-loop machine Σ_φ, the external input character ν must induce a stable transition from x^i to x^j as well as a stable transition from $x^{i'}$ to $x^{j'}$. Considering that these two transitions use the same external input character ν, the feedback function φ depends for these two transitions only on the state of Σ. In view of Proposition 3, this implies that the union $\rho_{ij}(\Sigma) \cup \rho_{i'j'}(\Sigma)$ must contain an implementable set. Recalling Definition 4 of compatible union and Proposition 5, this implies that we must have $\rho_{ij}(\Sigma) \sqcup \rho_{i'j'}(\Sigma) \neq N$. The same argument applies to all stable transitions of Σ' that are induced by the external input character ν, since, for a constant external input character, the feedback function φ is only a function of

the state of Σ. This argument leads us to a necessary and sufficient condition under which model matching by state feedback is possible. At this point, it is convenient to introduce the following quantity for a character $v \in A$:

$$\mathcal{F}(v|\Sigma') = \left\{(i,j) \in \{1,2,\ldots,n\} \times \{1,2,\ldots,n\} : v \in R^1_{ij}(\Sigma')\right\}. \tag{22}$$

Theorem 4: *Let $\Sigma = (A, X, f)$ and $\Sigma' = (A, X, s')$ be asynchronous machines, where Σ' is a stable state machine. Let $\rho(\Sigma)$ be the chain reachability matrix of Σ, let $R^1(\Sigma')$ be the one-step matrix of stable transitions of Σ', and, for an input character $v \in A$, let $\mathcal{F}(v|\Sigma')$ be given by (22). Then, the following two statements are equivalent.*

(i) *There is a static state feedback function $\varphi : X \times A \to A$ for which $\Sigma_\varphi = \Sigma'$.*
(ii) $\sqcup_{(i,j)\in\mathcal{F}(v|\Sigma')}\rho_{ij}(\Sigma) \neq N$ *for all $v \in A$.*

Proof: The fact that (i) implies (ii) is a consequence of the discussion preceding the statement of the theorem. Conversely, assume that (ii) is valid. For a character $v \in A$, denote

$$\sigma(\Sigma|\Sigma', v) := \bigsqcup_{(i,j)\in\mathcal{F}(v|\Sigma')} \rho_{ij}(\Sigma).$$

If $\sigma(\Sigma|\Sigma', v) \neq N$, then, by Theorems 1 and 2, there is a feedback function $\phi_v : X \to A$ for which the closed-loop machine Σ_{ϕ_v} implements all one-step stable transitions induced by the input character v on the model Σ'. We can then combine the functions ϕ_v for different input characters $v \in A$ into one partial function $\varphi : X \times A \to X : \varphi(x, v) \mapsto \phi_v(x)$ at all pairs $(x, v) \in X \times A$ for which $\mathcal{F}(v|\Sigma') \neq \varnothing$. Then, by construction, the closed-loop machine Σ_φ simulates all one-step stable transitions of the model Σ'. As a result, Σ_φ also implements any succession of stable transitions of Σ', and we have $\Sigma_\varphi = \Sigma'$. This shows that (ii) implies (i), and our proof concludes. □

It is worthwhile to consider a few special circumstance that may appear in Theorem 4. First, clearly, if $\mathcal{F}(v|\Sigma') = \varnothing$, then the character v is not a permissible input character of the model Σ', and hence v will never be applied to the closed-loop machine Σ_φ that simulates the model. As a result, there is no need to define the feedback function φ on any of the pairs $(x, v), x \in X$, as these pairs will never appear during the operation of the closed-loop machine Σ_φ.

Similarly, consider an integer $i \in \{1, 2, \ldots, n\}$ for which $(i, j) \notin \mathcal{F}(v|\Sigma')$ for all integers $j \in \{1, 2, \ldots, n\}$. Then, the pair (x^i, v) is never used by the model Σ'. Clearly, in such case, the feedback function φ does not need to be defined on the pair (x^i, v), as this pair will never be used during model matching.

The general process of constructing a feedback function for traditional model matching can be outlined as follows.

Procedure 1: Construction of a feedback function (traditional model matching).

Let $\Sigma = (A, X, f)$ and $\Sigma' = (A, X, s')$ be asynchronous machines with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the input alphabet $A = \{v^1, \ldots, v^m\}$, where Σ' is a stable state machine. Let $\rho(\Sigma)$ be the chain reachability matrix of Σ and let $R^1(\Sigma')$ be the one-step matrix of stable transitions of Σ'. Assume that condition (ii) of Theorem 4 is satisfied. Then, performing the following steps for every input character $v \in A$ results in a construction of a (partial) function $\varphi : X \times A \to A$ which, when used as a feedback function for Σ, yields $\Sigma_\varphi = \Sigma'$. The steps below are based on Theorem 4.

Step 1. Calculate the chain reachability matrix $\rho(\Sigma)$ of the controlled machine Σ, following the process described in Subsection 3.3.
Step 2. Calculate the one-step stable reachability matrix $R^1(\Sigma')$ of the model Σ' using (16).
Step 3. Set $\ell = 1$.
Step 4. Using (22), find the set of pairs $\mathcal{F}(v^\ell|\Sigma')$.
If $\mathcal{F}(v^\ell|\Sigma') = \varnothing$ and $\ell < m$, then replace ℓ by $\ell + 1$ and repeat from the beginning of Step 4.
If $\mathcal{F}(v^\ell|\Sigma') = \varnothing$ and $\ell = m$, then go to Step 8.
Step 5. If $\sqcup_{(i,j)\in\mathcal{F}(v^\ell|\Sigma')}\rho_{ij}(\Sigma) = N$, then model matching is not possible (Theorem 4). Terminate the procedure.
Step 6. Pick any member of $\sqcup_{(i,j)\in\mathcal{F}(v^\ell|\Sigma')}\rho_{ij}(\Sigma)$ and denote it by S^ℓ. Let $t(\ell)$ be the cardinality of S^ℓ and let

$$\{(x_{\ell 1}, u_{\ell 1}), (x_{\ell 2}, u_{\ell 2}), \ldots, (x_{\ell t(\ell)}, u_{\ell t(\ell)})\} \subseteq X \times A$$

be the elements of S^ℓ. Define the feedback function $\varphi : X \times A \to A$ on the subset $X \times \{v^\ell\}$ by setting

$$\varphi(x_{\ell i}, v^\ell) := u_{\ell i}, i = 1, 2, \ldots, t(\ell).$$

Step 7: If $\ell < m$, then replace ℓ by $\ell + 1$ and repeat from Step 4.
Step 8: Terminate the procedure. The resulting function φ achieves model matching $\Sigma_\varphi = \Sigma'$.

Needless to say, Procedure 1 is utilized only once during the life of a system, namely during the system's design stage. We demonstrate the construction of feedback functions in Example 15 below. Before that, we discuss a few features and special cases that help clarify several underlying issues.

5.2. Comparing the conditions for table model matching with the conditions for traditional model matching

To compare our discussion in this section to the discussion of Section 4, it is helpful to restate Theorem 3 in a form that more closely resembles Theorem 4. To this end, let Σ be an asynchronous machine with the state set X and the input alphabet A, and let Σ' be an asynchronous machine model with the same state set and input alphabet. A brief examination of Theorem 3 shows that, in our present notation, the machine Σ can match the model Σ' by table model matching if and only if $\rho_{ij}(\Sigma) \neq N$ whenever there is an input character $v \in A$ for which $(i, j) \in \mathcal{F}(v|\Sigma')$. It is worthwhile to restate Theorem 3 in these terms.

Theorem 5: *Let $\Sigma = (A, X, f)$ and $\Sigma' = (A, X, s')$ be asynchronous machines, where Σ' is a stable state machine. Let $\rho(\Sigma)$ be the chain reachability matrix of Σ, let $R^1(\Sigma')$ be the one-step matrix of stable transitions of Σ', and, for an input character $v \in A$, let $\mathcal{F}(v|\Sigma')$ be given by (22). Then, the following two statements are equivalent.*

(i) *Σ can match the model Σ' through table model matching.*
(ii) *$\rho_{ij}(\Sigma) \neq N$ whenever there is an input character $v \in A$ for which $(i, j) \in \mathcal{F}(v|\Sigma')$.*

In view of Definition 4 of the compatible union, condition (ii) of Theorem 4 is clearly violated if there is an input character $v \in A$ such that $\rho_{ij}(\Sigma) = N$ for a pair $(i,j) \in \mathcal{F}(v|\Sigma')$. By this observation, Theorems 5 and 4 lead to the following statement.

Corollary 1: *If traditional model matching by static state feedback is possible, then so is table model matching.*

Note, however, that the converse of Corollary 1 is not true in general: the feasibility of table model matching does not imply the feasibility of traditional model matching. Still, as the condition for table model matching (Theorem 3(ii)) is easier to check than the condition for traditional model matching (Theorem 4(ii)), it is helpful to check first whether the model Σ' can be matched by Σ through table model matching; if not, then traditional model matching will also not be possible. On the other hand, if table model matching is possible, then we can continue to check whether the condition for traditional model matching holds as well.

In the next subsection, we point out a special, but not uncommon, case in which the calculation of a feedback function φ that achieves traditional model matching can be simplified. Specifically, we refer to cases where there are similarities between the machine Σ and the model Σ'.

5.3. Exploiting similarities between the controlled machine and the model

Let Σ be an asynchronous machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the stable recursion function s, and let Σ' be a model with the same state set X and with the stable recursion function s'. As before, our objective is to find a feedback function $\varphi : X \times A \to A$ for which $\Sigma_\varphi = \Sigma'$. Now, let $v \in A$ be an input character and refer to the family $\mathcal{F}(v|\Sigma')$ of (22). Assume that there is a pair of integers $(i,j) \in \mathcal{F}(v|\Sigma')$ for which also

$$s(x^i, v) = x^j. \tag{23}$$

Then, the machine Σ has a stable transition that is identical to a stable transition of the model Σ'. This stable transition of Σ may originate from a chain of transitions through the states $x^i, x^i_1, \ldots, x^i_{k_i-1}, x^j$ of Σ; here, $x^i_1, \ldots, x^i_{k_i-1}$ are transient states that Σ passes on its way from a stable combination with the state x^i to a stable combination with the state x^j. In such case, the entry $\rho_{ij}(\Sigma)$ of the chain transition matrix includes the member

$$\{(x^i, v), (x^i_1, v), \ldots, (x^i_{k_i-1}, v), (x^j, v)\}. \tag{24}$$

Further, assume that there is another pair of integers $(i',j') \in \mathcal{F}(v|\Sigma')$ such that

$$s(x^{i'}, v) = x^{j'}.$$

Just as before, this pair may induce a chain of transitions in Σ, passing through states $x^{i'}, x^{i'}_1, \ldots, x^{i'}_{k_{i'}-1}, x^{j'}$. Then, the entry $\rho_{i'j'}(\Sigma)$ of the chain transition matrix includes the member

$$\left\{(x^{i'}, v), (x^{i'}_1, v), \ldots, (x^{i'}_{k_{i'}-1}, v), (x^{j'}, v)\right\}. \tag{25}$$

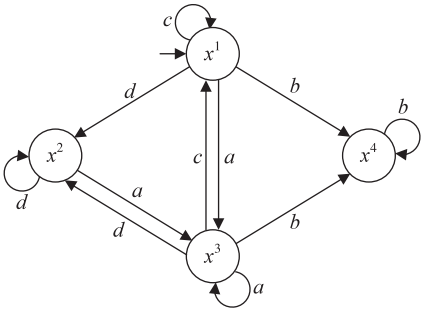

Figure 4. State flow diagram of Σ_2'.

As the members (24) and (25) both have the same input character v, they are consistent. Consequently,

$$\{(x^i, v), (x_1^i, v), \ldots, (x_{k_i-1}^i, v), (x^j, v)\} \cup \{(x^{i'}, v), (x_1^{i'}, v), \ldots, (x_{k_{i'}-1}^{i'}, v), (x^{j'}, v)\} \in \rho_{ij}(\Sigma) \sqcup \rho_{i'j'}(\Sigma). \quad (26)$$

The argument used to derive (26) leads to the following conclusion.

Proposition 8: *Let Σ and Σ' be asynchronous machines with the same state set $X = \{x^1, \ldots, x^n\}$. Let s be the stable recursion function of Σ, let $\rho(\Sigma)$ be the chain transition matrix, and let $\mathcal{F}(v|\Sigma')$ be given by (22). Assume that there is an input character $v \in A$ such that $s(x^i, v) = x^j$ for all pairs $(i, j) \in \mathcal{F}(v|\Sigma')$. Then, using the notation of (24), the following is true.*

$$\bigcup_{(i,j)\in\mathcal{F}(v|\Sigma')} \{(x^i, v), (x_1^i, v), \ldots, (x_{k_i-1}^i, v), (x^j, v)\} \in \bigsqcup_{(i,j)\in\mathcal{F}(v|\Sigma')} \rho_{ij}(\Sigma). \quad (27)$$

When the condition of Proposition 8 is satisfied, then the elements of the union on the left side of (27) represent valid pairs of the controlled machine Σ for which the stable state behaviour of Σ itself simulates the model Σ'. In such case, the external input character v of the closed-loop machine Σ_φ can be directly applied to the machine Σ; the feedback function φ can then serve as the identity function for this input character. In other words, we can set

$$\varphi(x^i, v) := v \text{ whenever } s(x^i, v) = x^j \text{ for all pairs } (i, j) \in \mathcal{F}(v|\Sigma'). \quad (28)$$

Of course, for input characters for which (28) is not valid, the feedback function φ must be constructed by following Procedure 1. If applicable, this observation may simplify the construction of the feedback function φ.

Example 15: Assume that the machine Σ of Example 1 must be adapted to yet another operating environment, where its response must conform to the asynchronous machine Σ_2' described by the state flow diagram of Figure 4. We show now that this adaptation can be accomplished by traditional model matching.

Table 6. The stable recursion function s_2'.

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

The stable recursion function s_2' of Σ_2' is described by Table 6 (note that Σ_2' is a stable state machine). Using the stable recursion function s_2', we can find the one-step matrix of stable transitions of Σ_2', which is given by (29).

$$R^1(\Sigma_2') = \begin{pmatrix} \{c\} & \{d\} & \{a\} & \{b\} \\ N & \{d\} & \{a\} & N \\ \{c\} & \{d\} & \{a\} & \{b\} \\ N & N & N & \{b\} \end{pmatrix}. \tag{29}$$

Comparing Tables 2 and 6, we can see that $s(x, v) = s_2'(x, v)$ for all valid pairs $(x, v) \in X \times \{a, b, c\}$. Consequently, in view of (28), we can set

$$\varphi(x, v) = v \text{ for all } (x, v) \in X \times \{a, b, c\}. \tag{30}$$

Thus, we only have to check the case of the input character d. For this input character, it follows from (29) that

$$\mathcal{F}(d|\Sigma_2') = \{(1, 2), (2, 2), (3, 2)\}.$$

Applying Theorem 4, we calculate the compatible union

$$\begin{aligned} \bigsqcup_{(i,j)\in\mathcal{F}(d|\Sigma_2')} \rho_{ij}(\Sigma) &= \rho_{12}(\Sigma) \sqcup \rho_{22}(\Sigma) \sqcup \rho_{32}(\Sigma) \\ &= \{(x^1, d), (x^2, d)\} \sqcup \{(x^2, d)\} \sqcup \{(x^3, c), (x^1, d), (x^2, d)\} \\ &= \{(x^1, d), (x^2, d), (x^3, c)\} \neq N. \end{aligned} \tag{31}$$

Combining (31) with (30) and Proposition 8, we conclude that $\sqcup_{(i,j)\in\mathcal{F}(v|\Sigma')}\rho_{ij}(\Sigma) \neq N$ for all $v \in A$. Consequently, it follows by Theorem 4 that there is a static state feedback function $\varphi : X \times A \to A$ for which $\Sigma_\varphi = \Sigma_2'$.

We turn now to the assembly of the feedback function $\varphi : X \times A \to A$. First, referring to (30), we can set

$$\varphi(x, v) := v \text{ for all states } x \in X \text{ and for all input characters } v \in \{a, b, c\}.$$

Further, for the input character d, we define φ according to the pairs that appear in (31), namely we set

$$\begin{aligned} \varphi(x^1, d) &:= d, \\ \varphi(x^2, d) &:= d, \\ \varphi(x^3, d) &:= c. \end{aligned}$$

This completes the construction of the feedback function φ in this case. With this feedback function, we obtain $\Sigma_\varphi = \Sigma_2'$.

Note that, in this case, Corollary 1 implies that there also exists a feedback function φ_t which, when applied to Σ, achieves table model matching of Σ_2'.

As we can see from Examples 14 and 15, it is often possible to achieve design objectives using static state feedback controllers. When this is possible, it facilitates a controller implementation that is simpler than the implementation of dynamic controllers employed by Murphy, Geng, and Hammer (2002, 2003) and Venkatraman and Hammer (2006b).

Example 16: In Example 14, we have introduced the model Σ_1' and we have seen that Σ can match Σ_1' via table model matching. Let us examine now whether Σ can also match the model Σ_1' by traditional model matching. To this end, recall that the one-step matrix of stable transitions $R^1(\Sigma_1')$ is given in (19), while the chain reachability matrix $\rho(\Sigma)$ is given in Example 12. Let us examine, for instance, the input character d. From (19) we can see that

$$\mathcal{F}(d|\Sigma_1') = \{(2,2),(3,2),(1,4),(4,4)\}.$$

Applying Theorem 4, we calculate the compatible union

$$\bigsqcup_{(i,j)\in\mathcal{F}(d|\Sigma_1')} \rho_{ij}(\Sigma) = \rho_{22}(\Sigma) \sqcup \rho_{32}(\Sigma) \sqcup \rho_{14}(\Sigma) \sqcup \rho_{44}(\Sigma)$$
$$= \{(x^2,d)\} \sqcup \{(x^3,c),(x^1,d),(x^2,d)\} \sqcup \rho_{14}(\Sigma) \sqcup \{(x^4,b),(x^4,d)\},$$

where $\rho_{14}(\Sigma)$ is given by (10). A brief examination of (10) shows that the pair (x^3,c) that appears in $\rho_{32}(\Sigma)$ is incompatible with all members of $\rho_{14}(\Sigma)$, since the latter all include either the pair (x^3,b) or the pair (x^3,d). Consequently,

$$\bigsqcup_{(i,j)\in\mathcal{F}(d|\Sigma_1')} \rho_{ij}(\Sigma) = N,$$

and hence, by Theorem 4, it is not possible for the machine Σ to match Σ_1' by traditional model matching. However, as we have seen in Example 14, the machine Σ can match Σ_1' via table model matching.

6. Conclusion and future research

Static state feedback controllers furnish the simplest means of controlling asynchronous sequential machines, since static controllers are represented by functions, rather than by machines. In practical terms, this means that static controllers can be implemented by logical gates; they require no memory elements.

In this paper, we discussed two methodologies for model matching by static state feedback: traditional model matching and table model matching. The second methodology is applicable under somewhat broader conditions than the traditional one, at the expense of a slightly more involved implementation. For both methodologies, we derived necessary and sufficient conditions for achieving model matching by static state feedback. Checking these conditions is undemanding, as they are based simply on counting the number

of members of certain sets that are derived from the given recursion functions of the controlled machine and the desired model.

When model matching by static state feedback is possible, we provided effective procedures for the construction of appropriate feedback functions. The execution of these procedures was demonstrated by computational examples.

Future research in this direction includes characterizing the conditions under which static state feedback controllers can be deployed as part of an adaptive control scheme for asynchronous sequential machines; the conditions under which static state feedback controllers can be used as part of a defensive mechanism against adversarial interventions in the operation of asynchronous machines; and applications of static controllers in other branches of the theory of control for asynchronous sequential machines.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

The work of J.-M. Yang was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning [number 2015R1A2A1A15054026], and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education [number 2015R1D1A1A01056764].

Notes on contributors

Jung-Min Yang is a professor at the School of Electronics Engineering, Kyungpook National University, Republic of Korea. He received his BS, MS and PhD degrees in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST) in 1993, 1995 and 1999, respectively. His research interests include corrective control of asynchronous sequential machines and control of complex Boolean networks.

Jacob Hammer is a professor of electrical and computer engineering at the University of Florida in Gainesville, Florida, USA. He received his BSc, MSc and DSc degrees from the Technion – Israel Institute of Technology in Haifa, Israel. His research interests are in the general area of control theory, including the control of linear and non-linear systems and the control of asynchronous sequential machines.

References

Barrett, G., and S. Lafortune. 1998. "Bisimulation, the Supervisory Control Problem, and Strong Model Matching for Finite State Machines." *Discrete Event Dynamic Systems: Theory and Applications* 8 (4): 377–429.

Geng, X., and J. Hammer. 2005. "Input/Output Control of Asynchronous Sequential Machines." *IEEE Transactions on Automatic Control* 50 (12): 1956–1970.

Hammer, J. 1994. "On Some Control Problems in Molecular Biology." In *Proceedings of the IEEE Conference on Decision and Control*, Lake Buena Vista, FL, 4098–4103.

Hammer, J. 1996. "On Corrective Control of Sequential Machines." *International Journal of Control* 65 (2): 249–276.

Kohavi, Z. 1978. *Switching and Finite Automata Theory*. 2nd ed. New York: McGraw-Hill.

Kumar, R., S. Nelvagal, and S. I. Marcus. 1997. "A Discrete Event Systems Approach for Protocol Conversion." *Discrete Event Dynamic Systems: Theory and Applications* 7 (3): 295–315.

Martin, A. J., and M. Nyström. 2006. "Asynchronous Techniques for System-on-Chip Design." *Proceedings of IEEE* 94 (6): 1089–1120.

Murphy, T. E., X. Geng, and J. Hammer. 2002. "Controlling Races in Asynchronous Sequential Machines." In *Proceeding of the IFAC World Congress*, Barcelona: Elsevier BV.

Murphy, T. E., X. Geng, and J. Hammer. 2003. "On the Control of Asynchronous Machines with Races." *IEEE Transactions on Automatic Control* 48 (6): 1073–1081.

Peng, J., and J. Hammer. 2010. "Input/Output Control of Asynchronous Sequential Machines with Races." *International Journal of Control* 83 (1): 125–144.

Peng, J., and J. Hammer. 2012. "Bursts and Output Feedback Control of Non-deterministic Asynchronous Sequential Machines." *European Journal of Control* 18 (3): 286–300.

Sparsø, J., and S. Furber. 2001. *Principles of Asynchronous Circuit Design – A Systems Perspective*. Dordrecht: Kluwer Academic.

Thistle, J. G., and W. M. Wonham. 1994. "Control of Infinite Behavior of Finite Automata." *SIAM Journal on Control and Optimization* 32 (4): 1075–1097.

Tinder, R. F. 2009. *Asynchronous Sequential Machine Design and Analysis: A Comprehensive Development of the Design and Analysis of Clock Independent State Machines and Systems*. San Mateo, CA: Morgan & Claypool.

Venkatraman, N., and J. Hammer. 2006a. "Controllers for Asynchronous Machines with Infinite Cycles." In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, Kyoto.

Venkatraman, N., and J. Hammer. 2006b. "On the Control of Asynchronous Sequential Machines with Infinite Cycles." *International Journal of Control* 79 (7): 764–785.

Venkatraman, N., and J. Hammer. 2006c. "Stable Realizations of Asynchronous Sequential Machines with Infinite Cycles." In *Proceedings of the Asian Control Conference*, 45–51. Bali, Indonesia.

Yang, J.-M. 2011. "Model Matching Inclusion for Input/State Asynchronous Sequential Machines." *Automatica* 47 (3): 597–602.

Yang, J.-M., and J. Hammer. 2008. "State Feedback Control of Asynchronous Sequential Machines with Adversarial Inputs." *International Journal of Control* 81 (12): 1910–1929.

Yang, J.-M., and J. Hammer. 2010. "Asynchronous Sequential Machines with Adversarial Intervention: The Use of Bursts." *International Journal of Control* 83 (5): 956–969.

Yang, J.-M., and S. W. Kwak. 2010. "Realizing Fault-Tolerant Asynchronous Sequential Machines Using Corrective Control." *IEEE Transactions on Control Systems Technology* 18 (6): 1457–1463.

Yevtushenko, N., T. Villa, R. K. Brayton, A. Petrenko, and A. L. Sangiovanni-Vincentelli. 2008. "Compositionally Progressive Solutions of Synchronous FSM Equations." *Discrete Event Dynamic Systems: Theory and Applications* 18 (4): 51–89.