Proceedings of the 2006 Asian Control Conference, Bali, Indonesia

Stable Realizations of Asynchronous Sequential Machines with Infinite Cycles

Niranjan Venkatraman and Jacob Hammer

Abstract—A control theoretic framework is developed for the representation of asynchronous sequential machines afflicted by infinite cycles. The framework includes a realization theory, and it lays a foundation for the derivation of state-feedback controllers that eliminate the effects of infinite cycles.

I. Introduction

ASYNCHRONOUS sequential machines form the building blocks of some of the fastest computing machines available. An infinite cycle is a common defect of an asynchronous machine, causing the machine to loop indefinitely among several of its states. Infinite cycles can occur as a result of malfunctions, design flaws, component failures, or implementation flaws. The present note develops a realization theory for asynchronous machines with infinite cycles. The resulting state representation expresses in a convenient and clear form the functional implications of infinite cycles. It lays the groundwork for the development of a general methodology for the design of feedback compensators that eliminate the effects of infinite cycles ([17]).

The process of developing a realization theory for asynchronous machines with infinite cycles involves a generalization of the concept of state. Recall that, in qualitative terms, an asynchronous machine has two kinds of states: stable (or persistent) states, i.e., states in which the machine can linger indefinitely, and unstable states - transient states through which the machine passes in quick succession. An infinite cycle represents a situation where the machine cycles quickly among several transient states. However, since the machine can linger indefinitely in an infinite cycle, consistency requires us to interpret an infinite cycle as a persistent, or "stable" state. This observation leads to the notion of a "generalized state", which is fundamental to the development of feedback controllers that eliminate the effects of infinite cycles ([17]). The mathematical description of the persistent features of an asynchronous machine's behavior is referred to as a "stable realization". In this note, we build a stable realization with "generalized states", where a generalized state represents either a stable state of the machine or an infinite cycle. Stable realizations underlie the design of controllers for asynchronous machines with infinite cycles.

The discussion of this paper is based on [17], and is a continuation of [6], [7], [8], [9], [10], [12], [13], [4], and [5]. It seems that the literature contains no reports regarding the use of control techniques to eliminate the effects of infinite cycles in an existing asynchronous sequential machine.

Studies dealing with other aspects of the control of discrete event systems can be found in [14], [2], [15], [1], [3], and others.

Niranjan Venkatraman is with the Department of Electrical Engineering, Northern Arizona University, Flagstaff, AZ 86011, USA (phone: 928-523-0373; e-mail: v.niranjan@ieee.org).

Jacob Hammer is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA (phone: 352-392-4934; e-mail: hammer@mst.ufl.edu).

II. Terminology and Background

For a finite non-empty alphabet A, let A^* be the set of all finite strings of characters of A, and let A^+ be the set of all non-empty strings in A^*. The *concatenation* of two strings $w_1, w_2 \in A^*$ is the string $w := w_2 w_1$, obtained by appending w_1 to the end of w_2 (in reverse order). A *partial function* $f : S_1 \to S_2$ is a function whose domain is a subset of S_1.

An asynchronous machine is a sextuple $\Sigma = (A,Y,X,x_0,f,h)$, where A, Y, and X are nonempty finite sets, x_0 is the initial state, and $f : X\times A \to X$ and $h : X\times A \to Y$ are partial functions. Here, A is the *input alphabet,* Y is the *output alphabet*, and X is the set of *states;* f is the *recursion function* and h is the *output function*. A *valid pair* $(x,u) \in X\times A$ is a point at which the partial functions f and h are both defined.

The machine Σ starts from the initial state x_0 and accepts input strings of the form $u := u_0\, u_1 \ldots \in A^*$. In response, it generates a string of states $x_0x_1x_2 \ldots \in X^*$ and a string of output values $y_0y_1y_2 \ldots \in Y^*$ according to

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

The step counter k is incremented by one at every change of the input value or of the state value. The machine Σ is an *input/state machine* if $y_k = x_k$, $k = 0, 1, 2, \ldots$ An input/state machine Σ is represented by the triple (A,X,f), allowing for an arbitrary initial state.

A *stable combination* is a valid pair $(x,u) \in X\times A$ at which $f(x,u) = x$, i.e., the state x is a fixed point of f. An asynchronous machine lingers at a stable combination until an input change occurs. A pair (x,u) that is

not a stable combination is called a *transient pair*.

A transient pair (x,u) initiates a chain of transitions $x_1 = f(x,u)$, $x_2 = f(x_1,u)$, ..., where the input character u is kept fixed. This chain of transitions may or may not end. If it ends, then there is an integer $q \geq 1$ such that the state $x' := f(x_q,u)$ satisfies $x' = f(x',u)$, i.e., (x',u) is a stable combination. In such case, x' is called the *next stable state* of x with the input value u. If this chain of transitions does not terminate, then (x,u) is part of an *infinite cycle*.

The notion of next stable state leads to the *stable recursion function* $s : X\times A \to X$ of Σ, defined as follows. For a valid pair (x,u) of Σ that has a next stable state x', set $s(x,u) := x'$; leave s otherwise undefined. The *stable state machine* $\Sigma_{|s}$ induced by Σ is then the sextuple (A,X,Y,x_0,s,h), where s serves as the recursion function. The machine $\Sigma_{|s}$ describes the behavior of Σ as observed by a user: it ignores transients (which, ideally, occur in zero time).

If the input value of an asynchronous machine changes while the machine is undergoing transitions, then the response may become unpredictable, since the state of the machine at the time of the change is unpredictable. To avoid this uncertainty, asynchronous machines are normally operated in *fundamental mode*, where only one variable of the machine is allowed to change at a time. In fundamental mode, a change of the input value is allowed only while the machine is in a stable combination (e.g., [11]).

Fundamental mode operation is impossible when a machine is in an infinite cycle, since the machine never reaches a stable combination with its active input value. To take a machine out of an infinite cycle, the input value of the machine must be changed during the cycle. As it is not possible to predict at which state the machine is when such an input change is applied, the outcome of such an input change may be unpredictable.

For an integer $i \geq 1$ and a valid pair (x,u), the *i-th constant input iteration* $f^{\circ i}$ of f (with the input character u is defined by

$$f^{\circ 1}(x,u) := f(x,u),\ f^{\circ 2}(x,u) := f(f(x,u),u),\ ...,$$
$$f^{\circ i}(x,u) := f(f^{\circ i-1}(x,u),u),\ i = 2, 3, ...$$

III. Infinite Cycles

In an infinite cycle, an asynchronous machine moves indefinitely from one transient combination to another, while the input character is kept constant. Consider a machine Σ with the state set $X = \{x^1, x^2, ..., x^n\}$ and the input alphabet A. Assume that Σ has an infinite cycle χ that involves p states, say the states $x^{j_1}, x^{j_2}, ..., x^{j_p} \in X$, and the input character $a \in A$. The infinite cycle then functions according to the recursion

$$x^{j_{k+1}} = f(x^{j_k},a),\ k = 1, ..., p-1,$$
$$x^{j_1} = f(x^{j_p},a). \qquad (1)$$

We denote this infinite cycle by

$$\chi = \{x^{j_1}, x^{j_2}, ..., x^{j_p}; a\}.$$

The *input character* of χ is a, and the *state set* of χ is

$$X(\chi) := \{x^{j_1}, ..., x^{j_p}\} = \{x^{j_1}, f(x^{j_1},a), ..., f^{\circ(p-1)}(x^{j_1},a)\}. \qquad (2)$$

The *length* ℓ of the infinite cycle χ is the number of distinct states it contains, i.e., $\ell = p$ in this case.

An infinite cycle $\chi = \{x; a\}$ of length 1 satisfies $x = f(x,a)$, i.e., it forms a stable combination. *Unless specifically stated otherwise, the term "infinite cycle" is used exclusively for infinite cycles of length greater than 1.* Different infinite cycles associated with the same input character must have disjoint sets of states, as follows.

(3) LEMMA. A valid pair (x,a) of an asynchronous machine is a member of at most one infinite cycle.

Proof. According to (2), all states of an infinite cycle are generated from any pair (x,a) of the cycle by the list x, $f(x,a)$, $f^{\circ 2}(x,a)$, ... Thus, infinite cycles that include a common pair have identical states and input character, and whence form the same infinite cycle. ♦

A critical race describes a situation where the next state of an asynchronous machine is not unique ([16]). Formally, a *critical race pair* (x,u) of a machine $\Sigma = (A,Y,X,x_0,f,h)$ is a valid pair of Σ for which $f(x,u)$ is a subset of states, rather than a single state.

A. Finding all infinite cycles of a machine

The following matrix helps characterize all infinite cycles of an asynchronous machine ([17]).

(4) DEFINITION. Let $\Sigma = (A,X,x_0,Y,f,h)$ be an asynchronous machine with the state set $X = \{x^1, ..., x^n\}$, and let ε be a character not in A. Let U_{ij} be the set of input characters taking Σ in one step from x^j to x^i, i.e., $U_{ij} = \{u \in A : x^i \in f(x^j,u)\}$. The *one-step transition matrix* $\tau(f)$ of Σ is an $n\times n$ matrix, whose (i,j) entry is

$$\tau_{ij}(f) := \begin{cases} U_{ij} & \text{if } U_{ij} \neq \varnothing, \\ \varepsilon & \text{if } U_{ij} = \varnothing, \end{cases} \quad i, j = 1, ..., n. \qquad ♦$$

The matrix $\tau(f)$ helps visualize features of Σ.

(5) LEMMA. If Σ has no critical races, then no input character appears more than once in a column of $\tau(f)$.

Proof. Assume that an input character u appears in rows $p \neq q$ of column j of $\tau(f)$. Then $x^p \in f(x^j,u)$ and $x^q \in f(x^j,u)$, so that (x^j,u) is a critical race pair. ♦

We assume from now on that the machine Σ has no critical races. Still, critical races play an important role in our discussion, as they are often created when Σ is taken out of an infinite cycle. To examine the response of Σ over several steps, we define two operations on the one-step transition matrix. First, define a 'union' of characters:

$$\varepsilon \bigcup \varepsilon := \varepsilon,$$
$$u \bigcup \varepsilon := u \text{ for all } u \in A.$$

Note that ε behaves similarly to a 'zero' under this operation. Next, define an operation of *multiplication* over

the set $A' := A \cup \varepsilon$:

$$u \circ u := u,$$
$$u \circ u' := \varepsilon \text{ for all elements } u \neq u' \in A'. \quad (6)$$

For subsets $\{a_1, a_2, ..., a_q\}, \{b_1, b_2, ..., b_r\} \subset A'$, the multiplication is defined pairwise over all combinations

$$\{a_1, a_2, ..., a_q\} \circ \{b_1, b_2, ..., b_r\} = \{a_i \circ b_j\}_{i=1, ..., q,\ j=1, ..., r} \quad (7)$$

Let C and D be n×n matrices whose entries are subsets of A'. Their *union* is defined entrywise by

$$(C \cup D)_{ij} := C_{ij} \cup D_{ij},\ i, j = 1, ..., n,$$

reminiscent of the addition of numerical matrices. The next operation is reminiscent of numerical matrix multiplication.

$$(C \circ D)_{ij} := \bigcup_{k=1, ..., n} C_{ik} \circ D_{kj} \text{ for all } i, j = 1, ..., n. \quad (8)$$

(9) EXAMPLE.

$$C = \begin{pmatrix} a & \{a,b\} \\ \{b,c\} & b \end{pmatrix};\ D = \begin{pmatrix} b & c \\ a & a \end{pmatrix},\ C \circ D = \begin{pmatrix} a & a \\ b & c \end{pmatrix}. \ \blacklozenge$$

Multiplication is equivalent to constant input iteration:

(10) LEMMA. Let $f : X \times A \to X$ be a recursion function with the one-step transition matrix $\tau(f)$, and let $\tau(f^{\circ r})$ be the one-step transition matrix of the constant input iteration $f^{\circ r}$, $r = 2, 3, ...$ Then, $\tau(f^{\circ r}) = \tau(f^{\circ(r-1)}) \circ \tau(f)$.

Before stating the proof, we provide an example.

(11) EXAMPLE. Consider a machine Σ with the input set $A = \{a, b, c\}$, the state set $X = \{x^1, x^2, x^3\}$, and the transition function f. The state transition table of Σ is

	a	b	c
x^1	x^1	x^3	x^1
x^2	x^2	x^3	x^3
x^3	x^2	x^2	x^3

From the table, the one-step transition matrix of Σ is

$$\tau(f) = \begin{pmatrix} \{a,c\} & \varepsilon & \varepsilon \\ \varepsilon & a & \{a,b\} \\ b & \{b,c\} & c \end{pmatrix}. \quad (12)$$

A direct calculation yields

$$\tau(f^{\circ 2}) = \tau(f) \circ \tau(f) = \begin{pmatrix} \{a,c\} & \varepsilon & \varepsilon \\ b & \{a,b\} & a \\ \varepsilon & c & \{b,c\} \end{pmatrix}. \ \blacklozenge \quad (13)$$

Proof (Lemma 10). We demonstrate the case $r = 2$. Let $x^1, ..., x^n$ be the states of Σ. By (8), (7), and (6), the following two statements are equivalent for all $u \in A$.

(i) The (i,j) entry of $\tau(f) \circ \tau(f)$ includes u.

(ii) There is an integer $k \in \{1, ..., n\}$ for which $u \in \tau_{ik}(f)$ and $u \in \tau_{kj}(f)$.

When (ii) holds, $x^i = f(x^k,u)$ and $x^k = f(x^j,u)$, so that $x^i = f(f(x^j,u),u) = f \circ f(x^j,u) = f^{\circ 2}(x^j,u)$; whence, (ii) implies that

(iii) $u \in \tau_{ij}(f^{\circ 2})$.

Conversely, if (iii) holds, it follows from the definition of $\tau(f^{\circ 2})$ that $x^i = f^{\circ 2}(x^j,u) = f(f(x^j,u),u)$. Setting $x^k := f(x^j,u)$, we have $x^i = f(x^k,u)$, which implies (ii). Thus, (ii) and (iii) are equivalent. Consequently, (i) and (iii) are equivalent, and whence $\tau(f^{\circ 2})$ and $\tau(f) \circ \tau(f)$ have the same entries. ♦

Denoting $\tau^{\circ p}(f) := \tau(f) \circ \tau(f) \circ ... \circ \tau(f)$ for an integer $p \geq 1$, we obtain from Lemma 10

(14) COROLLARY. $\tau(f^{\circ p}) = \tau^{\circ p}(f)$ for all integers $p \geq 1$. ♦

The following statement includes infinite cycles of length 1 (i.e., stable combinations).

(15) PROPOSITION. Let Σ be an asynchronous machine with the recursion function f, the state set $X = \{x^1, ..., x^n\}$, and the input alphabet A. The following three statements are equivalent for every valid pair $(x^i,u) \in X \times A$.

(a) There is an integer $q \geq 1$ for which u is included in the (i,i) entry of the matrix $\tau^{\circ q}(f)$.

(b) (x^i,u) is part of an infinite cycle, whose length ℓ is an integer divisor of q.

(c) $x^i = f^{\circ q}(x^i,u)$.

Proof. By (a) and Corollary 14

$$x^i = f^{\circ q}(x^i,u). \quad (16)$$

Hence, the set E of integers $p \geq 1$ for which $x^i = f^{\circ p}(x^i,u)$ is not empty. Let ℓ be the smallest integer in E. Then, $x^i = f^{\circ \ell}(x^i,u)$. Using the integer division algorithm, write $q = a\ell + r$, where $0 \leq r < \ell$. Substituting into (16) yields $x^i = f^{\circ q}(x^i,u) = f^{\circ r}(f^{\circ a\ell}(x^i,u),u) = f^{\circ r}(x^i,u)$, so that $r \in E$. Considering that $r < \ell$ and that ℓ is the smallest member of E, we conclude that $r = 0$; whence, (a) implies (b).

If (b) is valid, then $q = a\ell$ for an integer $a \geq 1$. Thus, $x^i = f^{\circ \ell}(x^i,u) = [f^{\circ \ell}]^{\circ a}(x^i,u) = f^{\circ a\ell}(x^i,u) = f^{\circ q}(x^i,u)$, implying (c).

Finally, assume that (c) is valid, so that $x^i = f^{\circ q}(x^i,u)$ for some integer $q \geq 1$. Then, $u \in \tau_{ii}(f^{\circ q})$ by Definition 4, and whence $u \in (\tau^{\circ q}(f))_{ii}$ by Corollary 14, implying (a). ♦

We find now all states of an infinite cycle from one state.

(17) PROPOSITION. Assume that Σ has an infinite cycle χ of length $\ell > 1$ with the input character u, and let x^{j0} be a state of χ. The other states $x^{j1}, x^{j2}, ..., x^{j\ell-1}$ of χ can be found recursively as follows: having found the state x^{ji}, the index j_{i+1} of the next state x^{ji+1} is the number of the row in which the character u appears in column j_i of $\tau(f)$, $i = 0, ..., \ell-2$.

Proof. Let f be the recursion function of Σ. By (1), we have $x^{ji+1} = f(x^{ji},u)$, $i = 0, ..., \ell-2$. Hence, by Definition 4, the character u appears in position (j_{i+1}, j_i) of $\tau(f)$. ♦

The next algorithm finds all infinite cycles of a machine

Σ. (#S is the number of elements of a set S.)

(18) ALGORITHM. Let $\tau(f)$ be the one step transition matrix of a machine Σ. The steps below are performed individually for each input character $u \in A$.

Step 1: Let $\Delta_1(u)$ be the set of all states of Σ corresponding to diagonal entries of $\tau(f)$ that include u. Set $\delta_1(u) := \Delta_1(u)$.

Step 2: For $i \geq 2$, let $\Delta_i(u)$ be the set of all states of Σ corresponding to diagonal entries of $\tau^{\circ i}(f)$ that include u. Define the difference set

$$\delta_i(u) := \Delta_i(u) \setminus \bigcup_{1 \leq j \leq i-1} \Delta_j(u). \qquad (19)$$

Step 3: Recall that n is the number of states of Σ. If

$$i+1 > n - \Sigma_{j=1}^{i} \#\delta_j(u), \qquad (20)$$

then the algorithm terminates for the character u. Otherwise, repeat from Step 2, using i+1 for i. ♦

As indicated in Theorem 22 below, the set $\delta_i(u)$ of Algorithm 18 consists of all states of Σ that are included in infinite cycles of length i with the input character u.

(21) EXAMPLE. We demonstrate Algorithm 18 on the machine Σ of Example 11, using the input character b. The one-step transition matrix $\tau(f)$ of Σ is given by (12).

Step 1. In view of (12), the character b does not appear on the main diagonal of $\tau(f)$, so $\Delta_1(b) = \delta_1(b) = \varnothing$.

Step 2. Since $i < 2$, skip to Step 3.

Step 3. Inequality (20) becomes $2 > 3 - 0$, which is false; consequently, go to Step 2 with $i := 2$.

Step 2. In view of (13), there are two occurrences of the input character b on the main diagonal entry of $\tau(f^{\circ 2})$: in positions (2,2) and (3,3). Consequently, $\Delta_2(b) = \{x^2, x^3\}$.

Step 2. $\delta_2(b) = \Delta_2(b) \setminus \Delta_1(b) = \{x^2, x^3\}$.

Step 3. Inequality (20) becomes here $3 > 3 - 2 = 1$, which is true. This terminates the algorithm for b.

Repeating Algorithm 18 for the other input characters, one obtains $\delta_1(a) = \{x^1, x^2\}$, $\delta_1(b) = \varnothing$, $\delta_2(b) = \{x^2, x^3\}$, and $\delta_1(c) = \{x^1, x^3\}$. By Theorem 21, the states x^1 and x^2 form cycles of length 1 (i.e., stable combinations) with the input character a; the states x^2 and x^3 form a cycle of length 2 with the input character b; and the states x^1 and x^3 form stable combinations with the input character c. ♦

(22) THEOREM. Let $i \geq 1$ be an integer. In the notation of Algorithm 18, the following are true.

(a) The set $\delta_i(u)$ consists of all states that are members of infinite cycles of length i with the input character u.

(b) The machine Σ has exactly $\#\delta_i(u)/i$ infinite cycles of length i with the input character u.

Proof (by induction). Induction assumption: (a) is valid for $i = 1, ..., k$. Note that (19) can be rewritten as

$$\delta_{k+1}(u) := \Delta_{k+1}(u) \setminus \bigcup_{1 \leq j \leq k} \delta_j(u), \; k = 1, 2, ... \qquad (23)$$

Now, by Proposition 15, the set $\Delta_{k+1}(u)$ consists of all states of Σ that are included in infinite cycles with the input character u, whose length is an integer divisor of k+1. Applying the induction assumption, we conclude that the states in $\delta_{k+1}(u)$ have the following properties:

(i) As members of $\Delta_{k+1}(u)$, they are states of infinite cycles with the input character u, and the length of these infinite cycles is an integer divisor of k+1.

(ii) They are not involved in cycles whose length is k or less, since the states $\bigcup_{1 \leq j \leq k} \delta_j(u)$ are excluded.

Thus, (a) is valid for $i = k+1$, proving (a) by induction. When (20) is satisfied, there are not enough states left for a new infinite cycle of length i+1; hence Step 3 of Algorithm 18. Finally, (b) is implied by (a) and Lemma 3. ♦

The following finds all infinite cycles of a machine.

(24) ALGORITHM. In the notation of Algorithm 18, all infinite cycles of length i with the input character u are found as follows.

Step 0. If $\delta_i(u) = \varnothing$, then Σ has no infinite cycles of length i with the input character u, and the algorithm terminates. Otherwise, set $d_0 := \delta_i(u)$ and $j := 0$.

Step 1. Pick a state x^{j_0} from d_j. Using the procedure of Proposition 17, find the remaining states x^{j_1}, x^{j_2}, ..., $x^{j_{i-1}}$ of the infinite cycle. Define $d_{j+1} := d_j \setminus \{x^{j_0}, x^{j_1}, x^{j_2}, ..., x^{j_{i-1}}\}$.

Step 2: If $d_{j+1} = \varnothing$, the algorithm terminates. Otherwise, repeat from Step 1, replacing j by j+1. ♦

(25) EXAMPLE. We demonstrate Algorithm 24 on the machine Σ of Example 21 for the input character b.

Step 0. $\delta_1(b) = \varnothing$, so there are no stable combinations with the input character b.

Step 0. $\delta_2(b) = \{x^2, x^3\}$, so there are infinite cycles of length 2 with the input character b. Set $d_0 := \delta_2(b)$ and $j := 0$.

Step 1. Pick the state x^2 from d_0. By Proposition 17, the state x^3 belongs to the same infinite cycle, yielding the infinite cycle $\{x^2, x^3; b\}$. As $d_1 := d_0 \setminus \{x^2, x^3\} = \varnothing$, the algorithm terminates for the input character b. ♦

IV. Stable-State Representations

Consider an asynchronous machine $\Sigma = (A,X,Y,x_0,f,h)$ that has no infinite cycles. Then, every valid pair (x,u) of Σ has a next stable state x′, and we define a partial function $s : X \times A \to X$ by setting $s(x,u) := x'$ for every valid pair (x,u). The function s is the *stable recursion function* of Σ. When s is used as a recursion function, it induces the *stable-state machine* $\Sigma_{|s} = (A,X,Y,x_0,s,h)$. The stable-state machine describes the persistent states of Σ, and hence describes the behavior of Σ as experienced by a user.

When Σ has infinite cycles, being in an infinite cycle is

clearly a persistent status of the machine, and this status is definitely experienced by the user. Thus, if the notion of stable-state machine is to remain true to its goal of representing the persistent features of Σ, it must include a representation of infinite cycles. This leads to the following generalization of the notion of a stable-state machine, which is critical to the development of control strategies.

(26) DEFINITION. Let Σ be an asynchronous machine with the state set $X = \{x^1, ..., x^n\}$, the input alphabet A, and $t > 0$ infinite cycles $\chi_1, ..., \chi_t$ of length greater than 1. With each infinite cycle χ_i, associate a new state x^{n+i}, called a *cycle state*. The set $\tilde{X} := \{x^1, ..., x^n, x^{n+1}, ..., x^{n+t}\}$ is *the augmented state set* of Σ. The elements of $\tilde{X}$ are the *generalized states* of Σ.

A pair $(x,u) \in \tilde{X}\times A$ is a *generalized valid pair* of Σ if one of the following holds: (i) $x \in X$ and (x,u) is a valid pair of Σ; or (ii) $x = x^{n+i}$ for an integer $i \in \{1, ..., t\}$ and u forms a valid pair with each state of the infinite cycle χ_i.

A pair $(x,u) \in \tilde{X}\times A$ is a *generalized stable combination* if (i) $x \in X$ and (x,u) is a stable combination of Σ; or if (ii) $x = x^{n+i}$ for an integer $i \in \{1, ..., t\}$ and u is the input character of the infinite cycle χ_i. ♦

Thus, a generalized stable combination includes every persistent status: a stable state or an infinite cycle.

(27) EXAMPLE. For the machine Σ of Example 25, we associate the cycle state x^4 with the (only) infinite cycle $\chi^1 := \{x^2, x^3; b\}$. The augmented state set is then $\tilde{X} = \{x^1, x^2, x^3, x^4\}$. The generalized stable combinations are (x^4,b), (x^1,a), (x^2,a), (x^1,c), and (x^3,c). ♦

We introduce now a new recursion function for $\Sigma = (A,Y,X,x_0,f,h)$. Let $\chi_1, ..., \chi_t$ be the infinite cycles of Σ, and let $\tilde{X} = \{x^1, ..., x^{n+t}\}$ be its generalized state set. Using the stable recursion function s, define a partial function $s^e : X\times A \to \tilde{X}$ over all valid pairs $(x,u) \in X\times A$ of Σ:

$$s^e(x,u) = \begin{cases} s(x,u) & \text{if } (x,u) \text{ has a next stable state,} \\ x^{n+i} & \text{if } (f(x,u),u) \text{ is a pair of the infinite cycle } \chi_i. \end{cases}$$

Recall that $X(\chi)$ is the set of states included in an infinite cycle χ, and let u be an input character that forms valid pairs with all states of χ. Denote by $s^e[X(\chi),u]$ the image of the set $X(\chi)\times u$ through s^e, namely,

$$s^e[X(\chi),u] := \{x' \in \tilde{X} : x' = s^e(x,u) \text{ and } x \in X(\chi)\}.$$

Note that $s^e[X(\chi),u]$ can be one or more states, depending on χ and on u. The next notion is critical to our discussion. It describes a function whose values are subsets of $\tilde{X}$. When a value is a single element x, we ignore the distinction between the element $x \in \tilde{X}$ and the subset $\{x\} \subset \tilde{X}$.

(28) DEFINITION. In the notation of the earlier paragraph, the *generalized stable recursion function* s of Σ is defined over all generalized valid pairs $(x,u) \in \tilde{X}\times A$ by

$$s(x,u) := \begin{cases} s^e(x,u) & \text{if } x \in X, \\ s^e[X(\chi_i),u] & \text{if } x = x^{n+i},\ i = 1, ..., t. \end{cases}$$

The *generalized stable-state machine* $\Sigma_{|s} = (A,\tilde{X},s)$ induced by Σ is an input/state machine with the state set $\tilde{X}$ and the recursion function s.

A state $x' \in \tilde{X}$ is *stably reachable* from a state $x \in \tilde{X}$ if there is a input string $u \in A^+$ for which $x' \in s(x,u)$. ♦

The generalized stable transition function describes the behavior of Σ as experienced by a user, i.e., the persistent aspects of the machine's response.

(29) EXAMPLE. For the machine Σ of Example 27, the generalized stable state machine is described by

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

The state x^4 represents the infinite cycle. ♦

A. Fundamental mode and stable reachability

As discussed earlier, fundamental mode operation is impossible for machines with infinite cycles, since exiting an infinite cycle requires an input change while the cycle is in progress. The following is the closest alternative.

(30) DEFINITION. An asynchronous machine Σ is operating in *semi-fundamental mode* if it operates in fundamental mode when not in an infinite cycle. ♦

In the generalized stable state machine $\Sigma_{|s}$, an infinite cycle χ of Σ is represented by a stable combination (x,u), where x is the cycle state corresponding to χ and u is the input value of χ. Accordingly, the following is valid.

(31) THEOREM. Semi-fundamental mode operation of a machine Σ is equivalent to fundamental mode operation of its generalized stable state machine $\Sigma_{|s}$. ♦

Thus, working with generalized stable state machines assures semi-fundamental mode operation, and stable reachability characterizes the possibilities of moving from one persistent state to another in semi-fundamental mode. This observation brings to light the practical significance of generalized stable realizations. To perform computations related to stable reachability, we use the following matrix.

(32) DEFINITION. Let Σ be an asynchronous machine with the generalized stable state machine $\Sigma_{|s} = (A,\tilde{X},s)$, where $\tilde{X} = \{x^1, ..., x^{n+t}\}$. Denote by $s^*(x^i,x^j)$ the set of all input characters $u \in A$ for which $x^i \in s(x^j,u)$, and let N be a character not included in the alphabet A. Then, the *matrix of one-step generalized stable transitions* $R(\Sigma_{|s})$ is an $(n+t)\times(n+t)$ matrix whose (i,j) entry is given by

$$R_{ij}(\Sigma_{|s}) = \begin{cases} s^*(x^i,x^j) \text{ if } s^*(x^i,x^j) \neq \varnothing, \\ N \text{ otherwise, } i, j = 1, ..., n+t. \end{cases} \quad \blacklozenge$$

In $R(\Sigma_{|s})$, the (i,j) entry, if not N, consists of all (single) input characters that take $\Sigma_{|s}$ from $x^j \in \tilde{X}$ to a generalized stable combination with $x^i \in \tilde{X}$. An (i,j) entry of N indicates that $\Sigma_{|s}$ cannot be driven from x^j to a generalized stable combination with x^i by applying a single input character.

(33) EXAMPLE. For the machine of Example 29,

$$R(\Sigma_{|s}) = \begin{pmatrix} \{a,c\} & N & N & N \\ N & a & a & a \\ N & c & c & c \\ b & b & b & b \end{pmatrix}. \blacklozenge \qquad (34)$$

To determine the reachability properties of Σ in semi-fundamental mode, we need to introduce special operations on the matrix $R(\Sigma_{|s})$ (compare to [12] and [13]). Let w_i be a subset of the set of strings A^* or be the character N, i = 1, 2. The operation $\bigcirc\!\!\!\!\cup$ of *unison* is

$$w_1 \bigcirc\!\!\!\!\cup\, w_2 := \begin{cases} w_1 \cup w_2 \text{ if } w_1 \subset A^* \text{ and } w_2 \subset A^*, \\ w_1 \text{ if } w_1 \subset A^* \text{ and } w_2 = N, \\ w_2 \text{ if } w_1 = N \text{ and } w_2 \subset A^*, \\ N \text{ if } w_1 = w_2 = N. \end{cases}$$

Here, N is treated like the empty set it represents. The *unison* $C := A \bigcirc\!\!\!\!\cup\, B$ of two n×n matrices A and B is defined entrywise by $C_{ij} := A_{ij} \bigcirc\!\!\!\!\cup\, B_{ij}$, i, j = 1, ..., n.

Concatenation of strings $w_1, w_2 \in A^* \cup N$ is defined by

$$\text{conc}(w_1,w_2) := \begin{cases} w_2w_1 \text{ if } w_1, w_2 \in A^*, \\ N \text{ if } w_1 = N \text{ or } w_2 = N. \end{cases}$$

For subsets $W = \{w_1, w_2, \ldots, w_q\}$ and $V = \{v_1, v_2, \ldots, v_r\}$, whose elements are either words of A^* or the character N,

$$\text{conc}(W,V) := \bigcirc\!\!\!\!\cup_{\substack{i=1,...,q \\ j=1,...,r}} \text{conc}(w_i,v_j);$$

the result is either a subset of A^* or the character N.

Next, we define an operation reminiscent of matrix multiplication. Let C and D to be two n×n matrices whose entries are either subsets of A^* or the character N. The *product* $Z := CD$ is an n×n matrix with the entries

$$Z_{ij} := \bigcirc\!\!\!\!\cup_{k=1}^{n} \text{conc}(C_{ik},D_{kj}), \; i,j = 1, \ldots, n.$$

Using this product, define the powers

$$R^q(\Sigma_{|s}) := R^{q-1}(\Sigma_{|s})R(\Sigma_{|s}), \; q = 2, 3, ...$$

If not N, the (i,j) entry of $R^q(\Sigma_{|s})$ is the set of all input strings that may take x^j to a generalized stable combination with x^i in exactly q generalized stable transitions (the outcome of these transitions may not be deterministic). If the (i,j) entry is N, then it is not possible to reach from x^j to a generalized stable combination with x^i in exactly q generalized stable transitions. Define

$$R^{(q)}(\Sigma_{|s}) := \bigcirc\!\!\!\!\cup_{p=1...q} R^p(\Sigma_{|s}), \; q = 2, 3, ... \qquad (35)$$

By construction, if not N, the (i,j) entry of $R^{(q)}(\Sigma_{|s})$ consists of all strings that may take the machine $\Sigma_{|s}$ from x^j to a generalized stable combination with x^i in q or fewer generalized stable transitions (the outcome of these transitions may not be deterministic). Stable reachability is then characterized as follows (the proof of the next statement is similar to that of [13, Lemma 3.9]).

(36) THEOREM. The following two statements are equivalent for an asynchronous machine Σ.

(i) The generalized state x^i is stably reachable from the generalized state x^j.

(ii) The (i,j) entry of $R^{(n+t-1)}(\Sigma_{|s})$ is not N. ♦

Thus, the matrix $R^{(n+t-1)}(\Sigma_{|s})$ characterizes all transitions between persistent states of Σ, namely, it characterizes all options of controlling the machine in semi-fundamental mode operation. It plays a critical role in the control of asynchronous machines with infinite cycles ([17] and [18]). The following statement follows from Theorem 36 and Lemma 5.

(37) PROPOSITION. Let Σ be an asynchronous machine with n states and t infinite cycles, and let $R(\Sigma_{|s})$ be its one-step matrix of stable transitions. Then, the following are equivalent for all input strings $u \in A^+$ and for all j = 1, ..., n+t.

(i) Applying u at the generalized state x^j results in a critical race.

(ii) The string u appears in more than one entry of column j of the matrix $R^{(n+t-1)}(\Sigma_{|s})$. ♦

V. Conclusion

A framework for the construction of stable realizations of asynchronous sequential machines was developed. In this framework, a "state" represents a persistent status of the machine; in particular, a state may represent a (traditional) stable state of the machine or an infinite cycle. This framework is essential for the design of state feedback controllers that eliminate the effects of infinite cycles on asynchronous sequential machines, as discussed in [17] and [18]).

VI. References

[1] G. Barrett and S. Lafortune, "Bisimulation, the Supervisory Control Problem, and Strong Model Matching for Finite State Machines," *Journal of Discrete Event Dynamic Systems*, vol. 8, no. 4, 1998, pp. 377–429.

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

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

[4] X. J. Geng and J. Hammer, "Input/output control of asynchronous sequential machines," IEEE Trans. Automatic Control (to appear).

[5] X. J. Geng and J. Hammer, "Asynchronous sequential machines: input/output control," *Proc. of the 12th Mediterranean Conf. on Control and Automation*, Kusadasi, Turkey, June 2004.

[6] J. Hammer, "On some control problems in molecular biology," *Proc. of the IEEE Conference on Proceedings and Control*, December 1994.

[7] J. Hammer, "On the modeling and control of biological signal chains," *Proc. IEEE Conference on Decision and Control*, December 1995.
[8] J. Hammer, "On the corrective control of sequential machines," *International J. of Control*, vol. 65, 1996, pp. 249-276.
[9] J. Hammer, "On the control of incompletely described sequential machines," *International J. of Control*, vol. 63, no. 6, 1996, pp. 1005-1028.
[10] J. Hammer, "On the control of sequential machines with disturbances," *International J. of Control*, vol. 67, no. 3, 1997, pp. 307-331.
[11] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill Book Company, New York, 1970.
[12] T. E. Murphy, X. J. Geng, and J. Hammer, "Controlling races in asynchronous sequential machines", *Proc. of the IFAC World Congress*, Barcelona, July 2002.
[13] T. E. Murphy, X. J. Geng, and J. Hammer, "On the control of asynchronous machines with races", *IEEE Trans. on Automatic Control*, vol. 48, no. 6, 2003, pp. 1073-1081.
[14] P. J. G. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, 1987, pp. 206-230.
[15] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," *SIAM J. on Control and Optimization*, vol. 32, no. 4, 1994, pp. 1075-1097.
[16] S. H. Unger "Hazards, critical races, and metastability," *IEEE Trans. on Computers*, vol. 44, no. 6, 1995, pp. 754-768.
[17] N. Venkatraman and J. Hammer, "On the control of asynchronous sequential machines with infinite cycles," International Journal of Control (to appear).
[18] N. Venkatraman and J. Hammer, "Controllers for asynchronous sequential machines with infinite cycles," Proceedings of the 2006 International Symposium on the Mathematical Theory of Networks and Systems (to appear).