International Journal of Control, 2016
Vol. 89, No. 1, 193–209, http://dx.doi.org/10.1080/00207179.2015.1064547

Automatic defensive control of asynchronous sequential machines

Jacob Hammer[*]

Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611-6130, USA

(*Received 11 March 2015; accepted 16 June 2015*)

Control theoretic techniques are utilised to develop automatic controllers that counteract robotic adversarial interventions in the operation of asynchronous sequential machines. The scenario centres on automatic protection against pre-programmed adversarial agents that attempt to subvert the operation of an asynchronous computing system. Necessary and sufficient conditions for the existence of defensive controllers that automatically defeat such adversarial agents are derived. These conditions are stated in terms of skeleton matrices – matrices of zeros and ones obtained directly from the given description of the asynchronous sequential machine being protected. When defensive controllers exist, a procedure for their design is outlined.

Keywords: adversarial intervention; feedback control; asynchronous sequential machines

1. Introduction

Adversarial agents that attempt to intervene in the operation of computing systems and networks have become alarmingly prevalent in recent years. The development of defensive mechanisms to counteract actions by such adversarial agents is increasingly turning into a prominent engineering enterprise. Often, such adversarial agents take the form of pre-programmed proxies implanted in computing systems, acting as adversarial controllers attempting to impair the stricken system and subject it to sinister objectives. Examples of such pre-programmed entities include computer viruses, computer worms, computer trojan horses, and other forms of malware. Contemporaneously with the rise in adversarial interventions, asynchronous sequential machines – platforms of high-speed computing – have been gaining in popularity to help fulfil the growing demand for computing power (Martin & Nyström, 2006; Sparsø & Furber, 2001; Tinder, 2009). As a result, it has become more urgent to develop a general theoretical framework for the analysis, design, and implementation of defensive mechanisms against adversarial interventions in the operation of asynchronous sequential machines.

In addition to their important role in computing systems and networks, asynchronous sequential machines also model the operation of signalling chains in molecular biology. Here, adversarial interventions take the form of biological viruses, senescence, and biochemical contaminants. Automatic controllers that can counteract such adversarial interventions can be implemented by inserting artificially produced genetic elements into affected cells.

Traditionally, the fight against adversarial software agents has been conducted mainly through the use of ad hoc techniques and 'tricks' in attempts to detect, misdirect, or outwit these software agents and neutralise their impact (e.g., Szor, 2005). The present paper takes a different approach, employing a systematic mathematical framework and utilising the tools of modern control theory to develop automatic means to counteract pre-programmed adversarial agents. The paper focuses on the development of automatic controllers that negate actions of pre-programmed adversarial agents attempting to interfere with the operation of an asynchronous sequential machine.

Specifically, we examine an asynchronous machine Σ tasked with reaching a prescribed beneficial target state, while a pre-programmed adversarial controller attempts to overtake Σ and drive it toward a harmful state. The situation is depicted in Figure 1, which shows Σ being operated by two controllers: an adversarial controller C_A and a defensive controller C_D. We assume that the state x of Σ is available as output, so that C_A and C_D are both state feedback controllers. The adversarial controller C_A may represent a virus or another adversarial entity that attempts to subvert Σ, while the defensive controller C_D represents a defensive mechanism designed to counteract actions of C_A and guide Σ to a desirable state. In the figure, C_A and C_D act through separate input channels; this includes the special case where both controllers act via the same input channel.

The controllers C_A and C_D of Figure 1 are asynchronous machines with no external input signals; they operate as pre-programmed autonomous state feedback controllers. Each

[*]Email: hammer@mst.ufl.edu

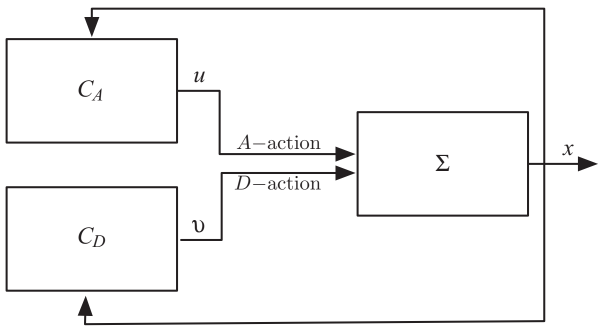

Figure 1. Adversarial/defensive control.

controller has its own performance objective. The composite machine described by Figure 1 is an asynchronous machine denoted by $\Sigma(C_A, C_D)$.

The controllers C_A and C_D operate in alternating turns; during its turn, each controller administers a string of commands to the machine Σ, while the other controller is at rest. The ultimate objective of each controller is to drive Σ into a *target set* of states that is specific to that controller. We denote by T_A the target set of the controller C_A and by T_D the target set of the controller C_D. As C_A and C_D have opposing objectives,

$$T_A \cap T_D = \varnothing,$$

the empty set. A controller *prevails* when Σ reaches one of its target states, at which point the control process of Figure 1 terminates. We concentrate on the following questions.

Problem 1.1: For the control configuration of Figure 1,

(i) Characterise the conditions under which each one of the controllers can prevail.
(ii) Characterise states of the machine Σ from which each one of the controllers is guaranteed to prevail.
(iii) If a controller cannot prevail, characterise the conditions under which it can prevent the other controller from prevailing.

In Sections 4 and 5, we derive necessary and sufficient conditions under which one of the controllers C_A or C_D can prevail. These conditions are stated in terms of certain skeleton matrices – matrices of zeros and ones derived from the given description of the controlled machine Σ. In Section 5, we also characterise states of Σ from which one of the controllers is guaranteed to prevail (when using an appropriate strategy), irrespective of actions taken by the other controller. When it is possible for one of the controllers to prevail, an appropriate controller design process is outlined in Section 4. The design process is based on a certain matrix of strings – the matrix of stable transitions, derived directly from the given description of the controlled machine Σ.

To guarantee deterministic behaviour, asynchronous machines must be operated under certain rules, collectively referred to as 'fundamental mode operation'. We discuss these rules in the remaining parts of this section.

1.1 Fundamental mode operation

An asynchronous sequential machine is a continuous-time nonlinear system with discrete and finite state, input, and output sets. Denoting the state set of the machine by X, its input set by U, its output set by V, and its initial state by x_0, the operation of the machine is governed by a *recursion function* $\psi: X \times U \to X$ and an *output function* $\eta: X \to V$ according to the recursion

$$\begin{aligned} x_{k+1} &= \psi(x_k, u_k), \\ v_k &= \eta(x_k),\ k = 0, 1, 2, \dots, \end{aligned}$$

where $x_0, x_1, x_2, \dots \in X$ form a string of states; $u_0, u_1, u_2, \dots \in U$ form a string of input characters; and $v_0, v_1, v_2, \dots \in V$ form the corresponding string of output characters. The integer k is the *step counter*.

The machine is in a *stable state* if $x = \psi(x, u)$; then, (x, u) is a *stable combination*. In a stable combination, the machine stays in its current state x until a new input takes effect.

When the pair (x, u) is not a stable combination, it is called a *transient combination*. A transient combination (x, u) gives rise to a chain of transitions

$$x_1 = \psi(x, u),\ x_2 = \psi(x_1, u),\ x_3 = \psi(x_2, u), \dots \quad (1.2)$$

with the same input character u. The duration of a transient step of an asynchronous machine is short and indeterminate. The machine progresses through a transient step at the speed limit of its components, which may depend on temperature, component condition, and other factors. Steps in a transition chain are not synchronous with any event. In an ideal asynchronous machine, a transient step is completed in zero time.

As a result of high speed and asynchrony, it is undesirable to change the input of an asynchronous machine during a chain of transitions: it is impossible to predict in what state the machine will be, when the input change takes effect. As an example, consider the transition chain of (1.2). Assume that one attempts to change the input character from u to u' right after the machine has reached the state x_1. This leads to a 'race': will the input change take effect before the machine moves to the next state x_2, resulting in the pair (x_1, u'); or will the input change take effect thereafter, resulting instead in the pair (x_2, u'), or may be the pair (x_3, u'), or may be a later pair? The answer to this question may depend on unpredictable conditions, such as temperature and component condition. As the response of the machine may be different in each case, an attempt to change the input during a chain of transitions may result in a non-deterministic response.

This argument gives rise to the notion of *fundamental mode operation* (e.g., Kohavi, 1978), whereby input changes are allowed only when an asynchronous machine is in a stable state. Fundamental mode operation assures a deterministic outcome.

1.2 Asynchronous trigger machines

An *asynchronous trigger machine* is an asynchronous machine that is operated by trigger inputs, namely, by inputs formed by short pulses that are, ideally, of zero duration (e.g., Kohavi, 1978). Each trigger carries a single input character; a trigger machine receives no input between consecutive triggers. In response to a trigger, the machine undergoes a string of transitions to reach a stable state, namely, a state at which the machine lingers until the next trigger input arrives. Such transitions occur very quickly (ideally, in zero time), and no input is present during transition.

Trigger machines are very common. Most asynchronous digital computers operate as trigger machines: a computer program invokes a momentary command, after which the computer proceeds into a chain of transitions to its next stable state. Trigger machines are somewhat easier to analyse, since the progression of the machine after a trigger is input free.

Considering the wide prevalence of asynchronous trigger machines in applications, we focus our attention in this paper exclusively on such machines. The results can be generalised to other types of asynchronous machines as well.

Assumption 1.3: *The machines Σ, C_A, and C_D of Figure 1 are asynchronous trigger machines.*

1.3 Asynchronous trigger machines with two inputs

The machine Σ of Figure 1 is an asynchronous input/state trigger machine with two inputs; it is characterised by a quintuple $\Sigma = (A, D, X, f, x_0)$, where A and D are non-empty input alphabets; X is the state set; f: $X \times (A \times D) \to X$ is a partial function serving as the *recursion function*; and x_0 is the initial state. The machine starts at the initial state x_0 and is driven by strings of trigger input pairs $(u_0, \upsilon_0)(u_1, \upsilon_1)(u_2, \upsilon_2)\ldots$, where $u_i \in A$ and $\upsilon_i \in D$, $i = 0, 1, 2, \ldots$ In response, Σ generates a string of states $x_0x_1x_2\ldots$ according to the recursion

$$\Sigma: \; x_{k+1} = f(x_k, (u_k, \upsilon_k)), \quad k = 0, 1, 2, \ldots \tag{1.4}$$

A triplet $(x, (u, \upsilon)) \in X \times (A \times D)$ is a *valid combination* if the function f is defined at it. A valid combination $(x, (u, \upsilon))$ is a *stable combination* if $x = f(x, (u, \upsilon))$, namely, if the machine Σ rests at the state x; in that case, x is a *stable state*. If $x \neq f(x, (u, \upsilon))$, then x is a *transient state*; an asynchronous machine leaves a transient state very quickly (ideally, in zero time).

Notation 1.5:

(i) Consider the asynchronous trigger machine $\Sigma = (A, D, X, f, x_0)$ of Figure 1. As transitions of an asynchronous trigger machine progress with no input after a trigger, it is convenient to add the extra character '$\neg$' to the alphabets A and D to denote the absence of a trigger. Then, a pair $(u, \neg) \in A \times D$ means that the character u is triggered into input A, while no trigger is applied to input D; similarly, $(\neg, \upsilon) \in A \times D$ means that the character υ is triggered into input D, while no trigger is applied to input A. The pair $(\neg, \neg)$ indicates that no trigger is applied to either input.

(ii) For a non-empty set S, we denote by S^+ the class of all non-empty strings of elements of S.

(iii) The state set of Σ is always taken as $X = \{x^1, x^2, \ldots, x^n\}$. A subset of states $S = \{x^{i_1}, x^{i_2}, \ldots, x^{i_q}\} \subseteq X$ is often identified with the set of integers $S = \{i_1, i_2, \ldots, i_q\}$.

1.4 Fundamental mode operation and trigger machines

In an asynchronous environment, one cannot assume that two independent triggers occur simultaneously. Indeed, when attempting simultaneous independent triggers, asynchrony entails that, inevitably, one trigger will appear before the other. Furthermore, it is impossible to predict which of the two triggers will appear first, since this may depend on erratic and unpredictable operating conditions, such as temperature or equipment condition. Considering that the outcome of sequential triggers may depend on the order in which the triggers appear, it follows that an attempt to induce simultaneous triggers may lead to an unpredictable outcome. Thus, to guarantee deterministic behaviour, simultaneous triggers are prohibited.

Assume now that the asynchronous trigger machine $\Sigma = (A, D, X, f, x_0)$ is at a stable state x, when it receives the input trigger $(u, \upsilon) \in (A \times D)$; here, one of u or υ must be the empty character $\neg$. Recalling that no inputs are received between triggers, this trigger results in a chain of transitions of the form

$$\begin{aligned} &x_1 = f(x, (u, \upsilon)), x_2 = f(x_1, (\neg, \neg)), \\ &x_3 = f(x_2, (\neg, \neg)), \ldots, \end{aligned} \tag{1.6}$$

which may or may not terminate. If this chain of transitions does not terminate, then Σ has an *infinite cycle*. In this paper, we concentrate exclusively on machines with no infinite cycles (the control of asynchronous machines with infinite cycles is considered in Venkatraman and Hammer (2006a, 2006b).

Assumption 1.7: *The asynchronous machine Σ of Figure 1 has no infinite cycles.*

As Σ has no infinite cycles, the chain of transitions (1.6) must terminate. Therefore, there is an integer $i \geq 1$ such that

$$x_i = f(x_i, (\neg, \neg)), \tag{1.8}$$

so that x_i is a stable state of Σ. We refer to x_i as the *next stable state* of the triplet $(x, (u, \upsilon))$. All other states in the transition chain (1.6) are *transient states*. In view of Assumption 1.7, every valid combination has a next stable state. Note that the machine Σ remains in a stable state until a new trigger input appears.

Considering that a trigger machine has no persistent inputs, it follows that, between triggers, the progression of an asynchronous trigger machine is entirely determined by the state of the machine. This implies the following feature.

Fact 1.9: *In an asynchronous trigger machine Σ, once a transition chain has been triggered, its progression is determined only by the states it encounters; the input plays no role after initiating the transition chain. Between triggers, a state x of Σ is either always a stable state or always a transient state.*

Using the notion of next stable state, we introduce a partial function s: $X \times (A \times D) \to X$ defined on all valid combinations of Σ by setting $s(x, (u, \upsilon)) := x'$, where x' is the next stable state of $(x, (u, \upsilon))$. The function s is called the *stable recursion function* of Σ.

A potential indeterminacy may arise when attempting to apply an input trigger, while a machine is in a transient state. As transition through a transient state is quick and unsynchronised, a trigger input aimed at a transient state may actually materialise after the machine has left that state. As a result, the actual state at which the trigger occurs is unpredictable, and hence so may be the outcome of such a trigger.

To prevent uncertainties and achieve deterministic behaviour, it is common practice to operate asynchronous trigger machines in fundamental mode operation, where no more than one input is triggered at a time, and input triggers appear only when machines are in stable states (e.g., Kohavi, 1978). When applied to the configuration of Figure 1, this implies the following.

Rule 1.10: *Fundamental mode operation. Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with two inputs. Assume that Σ is at a stable state x, when it is activated by a string of input triggers $(u_1, \upsilon_1)(u_2, \upsilon_2)\ldots \in (A \times D)^+$. Then, the following must be observed.*

(i) *Only one input of Σ may be triggered at a time, namely, either $u_i = \neg$ or $\upsilon_i = \neg$ for all $i = 1, 2, \ldots$*
(ii) *No input triggers appear, while Σ is in transition.*

Let s be the stable recursion function of Σ. When applying to Σ an input string

$$(u, \upsilon) = (u_1, \upsilon_1)(u_2, \upsilon_2)\ldots(u_j, \upsilon_j) \tag{1.11}$$

in fundamental mode operation, only one character of each input pair is triggered, and we must wait after each input trigger for Σ to reach its next stable state, before applying the next input trigger. This leads the machine to the final stable state

$$s(x, (u, \upsilon)) := s(\ldots s(s(x, (u_1, \upsilon_1), (u_2, \upsilon_2))\ldots(u_j, \upsilon_j)).$$

As transitions of asynchronous machines are quick (they occur, ideally, in zero time), the necessary waiting time between triggers is negligible. This also entails that users of asynchronous machines are aware only of stable states, since transients disappear quickly; the performance of an asynchronous machine is determined by its stable state behaviour.

A *stable transition* is a transition in fundamental mode operation from a stable state to a stable state. A state x' is *stably reachable* from a state x if there is a stable transition from x to x'. Applying Rule 1.10 to the configuration of Figure 1 leads to the following.

Rule 1.12: *Referring to the machines Σ, C_A, and C_D of Figure 1,*

(i) *No more than one of these machines may trigger at a time, and the trigger must be directed to a machine that is in a stable state; and*
(ii) *When one of these machines is in transition, the other two are in stable states.*

Rule 1.12 guarantees fundamental mode operation of the composite machine $\Sigma(C_A, C_D)$, thus assuring a deterministic response. The following terminology is used repeatedly in our discussion.

Definition 1.13: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the two inputs A and D, and let x and x' be two states of Σ.

(i) *A-action* is a string of triggers applied in fundamental mode operation to input A of Σ, with input D inactive.

(ii) *D-action* is a string of triggers applied in fundamental mode operation to input D of Σ, with input A inactive.
(iii) x' is *stably reachable* from x *by A-action* if there is an *A-action* that takes Σ from a stable state at x to a stable state at x'.
(iv) x' is *stably reachable* from x *by D-action* if there is a *D-action* that takes Σ from a stable state at x to a stable state at x'.

As mentioned before, users of an asynchronous machine are affected only by stable states, since transitions through transient states are quick. Thus, when controlling an asynchronous machine, only stable states matter for performance. In particular, members of the target sets T_A and T_D of Σ are all stable states.

1.5 Controller turns

In the configuration of Figure 1, we assume that it is in the best interest of both controllers to maintain a deterministic response of the closed-loop machine $\Sigma(C_A, C_D)$. As a result, the controllers C_A and C_D operate in alternating turns, complying with fundamental mode operation. During each turn, one of the controllers is active, while the other one rests in a stable state. The active controller attempts to drive the machine Σ in fundamental mode operation to one of that controller's target states. If such a target state is stably reachable through that controller's action, then the controller prevails, and the control process terminates. Otherwise, the active controller drives Σ to a stable state most favourable to its objectives and pauses; the other controller then activates to start its own turn.

To express a controller's objectives in quantitative terms, we assign a weight to each state of the machine Σ: members of the adversarial target set T_A have the lowest weight, while members of the defensive target set T_D have the highest weight. In each turn, the adversarial controller C_A takes Σ to a lowest weight state stably reachable by A-action, while C_D takes Σ to a highest weight state stably reachable by D-action.

This framework represents many common applications; consider, for example, a power utility operated by a computer control system. In somewhat oversimplified terms, at each attempt, an adversarial controller aims to take the system to a state of lowest power generation, while a defensive controller aims to take the system to a state of highest power generation.

In Sections 4 and 5, we characterise all possible outcomes of this turn-by-turn control process. Specifically, there are three possible outcomes: (1) One of the controllers prevails; (2) the composite machine $\Sigma(C_A, C_D)$ reaches a stable state that is not a target state of any controller; or (3) the composite machine $\Sigma(C_A, C_D)$ enters an infinite cycle. Assuming that both controllers employ the best control strategies, the outcome of the control process is pre-ordained by the stable reachability features of the controlled machine Σ. In Sections 4 and 5, we derive necessary and sufficient conditions for each possible outcome. These conditions are stated in terms of skeleton matrices of Σ – certain matrices of zeros and ones derived directly from the given recursion function of the machine Σ. Appropriate controller designs are also outlined.

Another interesting question is whether there are any states of the machine Σ from which one of the controllers can always prevail, irrespective of actions taken by the other controller. We characterise such states in Section 5. In the same section, we also characterise states of Σ from which one of the controllers can always block the other controller from prevailing. In all cases, the structure of appropriate controllers is outlined.

1.6 General background

The present paper is written within the framework for the control of asynchronous sequential machines of Murphy, Geng, and Hammer (2002, 2003), Geng and Hammer (2005), Venkatraman and Hammer (2006a, 2006b), Yang and Hammer (2008, 2010), Peng and Hammer (2010, 2012), Yang (2011), and Yang and Kwak (2010). In particular, the current study continues the work of Yang and Hammer (2010), where the protection of asynchronous machines against adversarial interventions is considered, with a focus on model matching in the face of unspecified adversarial interventions. The present study examines the design of pre-programmed automatic controllers that counteract the actions of pre-programmed adversarial agents. Protection against adversarial agents is important in many applications, including high-speed computing systems, asynchronous networks, computer controlled utilities, and other computer-controlled services, as well as in the fight against viruses and senescence in molecular biology. A byproduct of our discussion is also a methodology for the design of effective adversarial agents.

There is an extensive literature on the control of finite state sequential machines, including publications such as Thistle and Wonham (1994), Hammer (1994, 1996), Kumar, Nelvagal, and Marcus (1997), Barrett and Lafortune (1998), Di Benedetto, Sangiovanni-Vincentelli, and Villa (2001), Yevtushenko, Villa, Brayton, Petrenko, and Sangiovanni-Vincentelli (2008), the references cited in these publications, and many others. The studies mentioned in this paragraph do not address issues that are specific to the operation of asynchronous sequential machines, such as the distinction between stable and transient states or the requirement of fundamental mode operation.

The paper is organised as follows. Section 2 introduces the formalism necessary for handling asynchronous machines with multiple inputs. This formalism is further developed in Sections 3 and 4, where the notions of ‘matrix of stable transitions' and 'skeleton matrix' of Murphy et al. (2002, 2003) are adapted to asynchronous machines with multiple inputs. The resulting notions are then utilised in Sections 4 and 5 to derive necessary and sufficient conditions for one of the controllers of Figure 1 to prevail, as well as to outline procedures for controller design. An example runs through the entire paper to highlight notions and to demonstrate constructions.

2. Basic notions

2.1 State weights

Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine controlled by two asynchronous trigger machines C_A and C_D as depicted in Figure 1. Recall that the controllers C_A and C_D operate in alternating turns; a controller's turn starts after the other controller has stably reached a 'most favourable outcome'. The term 'most favourable outcome' is quantified through state weights, as follows.

Definition 2.1: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the target sets T_A, $T_D \subseteq X$. An integer valued function ω: $X \to Z$ is a *weight function* for Σ if

(i) T_A is the set of states at which ω is maximal; and
(ii) T_D is the set of states at which ω is minimal.

As we can see from Definition 2.1, the weight function assigns a *weight* – an integer value – to each state. The adversarial controller's objective is to take Σ to a lowest weight state, while the objective of the defensive controller is to take Σ to a highest weight state. Using the weight function, we can clarify the discussion of Section 1.5 in the following way.

Rule 2.2: *Let Σ be an asynchronous trigger machine with weight function ω, operated by the two controllers C_A and C_D of Figure 1. In its turn, each controller drives Σ to a stably reachable extremal weight state: C_A ends its turn at a lowest weight state of Σ that is stably reachable by A-action, while C_D ends its turn at a highest weight state of Σ that is stably reachable by D-action.*

Example 2.3: Consider a stable state asynchronous trigger machine $\Sigma = (A, D, X, s, x_0)$, where $A = \{a^1, a^2\}$, $D = \{d^1, d^2\}$, $X = \{x^1, x^2, x^3, x^4, x^5\}$, initial state $x_0 = x^1$, and stable recursion function s given by Table 1; the sixth column of the table is the weight function ω of Σ. As we can see from the table, the target sets are $T_A = \{x^3\}$ and $T_D = \{x^5\}$. Input combinations not listed in the table are forbidden.

Needless to say, at some point, the controlled machine Σ can have several stably reachable states of the same extremal weight. In the name of efficiency, a controller stops at the first extremal weight state it stably reaches. This leads to the following refinement of Rule 2.2.

Table 1. Stable transitions of Σ.

x^i	$(a^1, \neg)$	$(a^2, \neg)$	$(\neg, d^1)$	$(\neg, d^2)$	$\omega(x^i)$	Target set
x^1	x^1	x^4	x^1	x^2	4	
x^2	x^1	x^3	x^2	x^1	4	
x^3	x^1	x^1	x^4	x^5	1	$\in T_A$
x^4	x^4	x^1	x^1	x^2	3	
x^5	x^1	x^4	x^1	x^2	5	$\in T_D$

Rule 2.4: *Operating policy. Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine operated by two asynchronous trigger controllers – an adversarial controller C_A and a defensive controller C_D, as depicted in Figure 1. Let T_A be the adversarial target set, let T_D be the defensive target set, and let ω: $X \to Z$ be the weight function of Σ. Then, the controllers C_A and C_D operate in alternate turns as follows.*

(i) *Start: Σ, C_A, and C_D are in stable states at their initial conditions, when C_A acts first.*
(ii) *Turns: In its turn, each controller drives Σ in fundamental mode operation to an extremal weight stably reachable state, stopping and ending its turn at the first extremal weight state of Σ it stably reaches. Here, C_A employs A-action and C_D employs D-action; extremal weight is lowest weight for C_A and highest weight for C_D. Each controller lingers in a stable state during the other controller's turn.*
(iii) *Progression:*
  (a) *Each controller activates when the other controller has reached the end of a turn.*
  (b) *A controller forfeits its turn if all states of Σ it can stably reach are target states of the other controller.*
(iv) *Target states: both controllers remain in a stable state, once Σ has reached a target state.*

Note that, by Rule 2.4*(ii)*, if a controller starts its turn at a state x of Σ whose weight is equal to the appropriate extremal weight of any state stably reachable from x by that controller, then Σ will remain at x as the controller's turn ends.

Example 2.5: To demonstrate Rule 2.4, refer to Table 1 and recall that Σ has the initial state x^1. Then, in the initial turn, C_A takes Σ to x^4, as this is the lowest weight stably reachable state from x^1 by A-action. Then, in the next turn, the highest weight stably reachable states from x^4 by D-action are x^1 and x^2. It is up to the designer of the controller C_D to select which of these two transitions would be implemented. Note that a transition to x^2 would form a poor choice, since it would allow the controller C_A to prevail in its next turn. A transition to x^1 would lead the closed-loop machine $\Sigma(C_A, C_D)$ to an infinite cycle; although not ideal, this would be a better choice in this case.

A brief examination of Rule 2.4 shows that the following statement characterises the conditions under which the composite machine $\Sigma(C_A, C_D)$ of Figure 1 reaches a stable state at the end of the control process. Note that, if a stable state is reached, it is reached very quickly (ideally, in zero time), since the entire process constitutes a transient of the closed loop machine.

Proposition 2.6: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with weight function ω: $X \to Z$ and target sets T_A and T_D, and assume that Σ is controlled in compliance with Rule 2.4. Then, (i) and (ii) are equivalent.*

(*i*) *The control process terminates with Σ in a stable state at the state x.*
(*ii*) *x is a target state of Σ, or all states stably reachable from x by both A-action and D-action have the same weight as x.*

2.2 The local sink

As our previous discussion intimates, the sets of extremal weight states that are stably reachable from a given state play an important role. This leads us to the following notion.

Definition 2.7: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine, and let $x \in X$ be a stable state of Σ. Denote by $S(x, A)$ (respectively, $S(x, D)$) the set of all states that are stably reachable from x by A-action (respectively, by D-action). Then, the *local A-sink* $S_A(x)$ consists of the lowest weight states that are stably reachable from x by A-action, namely

$$S_A(x) := \{x' \in S(x, A) : \omega(x') \le \omega(x'') \text{ for all } x'' \in S(x, A)\}.$$

Similarly, the *local D-sink* $S_D(x)$ consists of the highest weight states that are stably reachable from x by D-action, namely,

$$S_D(x) := \{x' \in S(x, D) : \omega(x') \ge \omega(x'') \text{ for all } x'' \in S(x, D)\}.$$

Example 2.8: Referring to the machine Σ of Example 2.3, we can see from Table 1 that $S_D(x^4) = \{x^1, x^2\}$.

By using state feedback, one can determine whether or not Σ has reached a stable state at a member of the local sink, as follows.

Lemma 2.9: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with weight function ω, operated by the two controllers of Figure 1 in compliance with Rule 2.4. Then, it can be determined by state feedback whether (and at what state) Σ has reached the end of a controller turn.*

Proof: Assume by induction that the statement of the lemma is true for turn j, where $j \ge 0$ is an integer. Let x be the state of Σ at the end of turn j, where $x = x_0$ for the initial turn $j = 0$. Assume further that the next turn, turn $j + 1$, is a turn of the controller C_A. Then, by our induction assumption, Σ is in a stable state at a known state $x \in X$, when C_A starts its turn. As C_A and C_D both employ state feedback, the state x is known at both controllers. Consequently, the local sink $S_A(x)$ is known at both controllers, since the local sink is determined by x and the given recursion function f of Σ. According to Rule 2.4*(ii)*, the controller C_A takes Σ to a stable state at a member $x' \in S_A(x)$, terminating its turn at the first member of $S_A(x)$ it encounters. Using state feedback, C_D detects when Σ has reached a member of $S_A(x)$ and starts its turn. Hence, the lemma is valid for turn $j + 1$, when it is a turn of C_A. The case where turn $j + 1$ is a turn of C_D is similar. This concludes our proof. □

The next statement shows that the operating policy of Rule 2.4 guarantees fundamental mode operation.

Proposition 2.10: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with weight function ω, operated by the two controllers C_A and C_D of Figure 1 in compliance with Rule 2.4. Then, the composite machine $\Sigma(C_A, C_D)$ operates in fundamental mode.*

Proof: To show fundamental mode operation, we have to show that *(1)* fundamental mode operation is preserved during each controller turn; and that *(2)* fundamental mode operation is preserved when switching from one controller turn to the next. Now, *(1)* is valid by Rule 2.4*(ii)*. Furthermore, *(2)* is a consequence of Lemma 2.9 since, according to the lemma, each controller can determine by state feedback when the other controller has reached the end of a turn. This concludes our proof. □

Controllers C_A and C_D that control the machine Σ in compliance with Rule 2.4 can be constructed by the following process (compare to Murphy et al. (2002, 2003) and Geng and Hammer (2005)).

Construction 2.11: Building the controllers C_A and C_D: Referring to Figure 1, let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with stable recursion function s: $X \times (A \times D) \to X$, weight function ω: $X \to Z$, and target sets T_A and T_D. For a stable state $x \in X$, let $S_A(x)$ be the local A-sink of Σ and let $S_D(x)$ be the local D-sink.

The controller C_A of Figure 1 is an asynchronous trigger machine that consists of two parts: an observer $\mathcal{O}$ and an action part C_A^a. The observer $\mathcal{O}$ detects the end of a controller turn for both C_A and C_D turns. The action part C_A^a is activated by $\mathcal{O}$ at the end of a C_D turn; C_A^a generates the A-action input string that drives Σ in fundamental mode operation during the turn of C_A. The structure of the controller C_D is similar – it consists of the observer $\mathcal{O}$ and an action part C_D^a analogous to C_A^a.

Part I. Building the observer $\mathcal{O}$: Let χ_A and χ_D be two distinct characters not included in the sets A, D or X. We use these as output characters of $\mathcal{O}$: a trigger of χ_A by $\mathcal{O}$ activates C_A^a to start a C_A turn, while a trigger of χ_D by $\mathcal{O}$ activates C_D^a to start a C_D turn. The observer $\mathcal{O}$ has access to the state of Σ through state feedback; it is constructed by following the proof of Lemma 2.9. For a C_A turn that starts at the state $x \in X$, the observer $\mathcal{O}$ detects when Σ stably reaches a state of $S_A(x)$. Similarly, for a C_D turn that starts at x, the observer $\mathcal{O}$ detects when Σ stably reaches a state of $S_D(x)$.

The observer is implemented by a function $\phi\colon X \times X \times \{A, D, N\} \to X \times \{\chi_A, \chi_D\}\colon (x, z, \zeta) \mapsto (x', \chi)$, where x is the state of Σ at the start of a controller turn; z is the current state of Σ; the character ζ indicates which controller is currently active – C_A, C_D, or no controller is active; x' indicates the state of Σ at the end of a controller's turn; and χ is one of the trigger characters χ_A, χ_D activating the next controller action. The character ζ has three possible values: A indicates that C_A is active, D indicates that C_D is active, and N indicates that no controller is active.

Recalling that x_0 is the initial state of Σ, applying Rule 2.4, and using the symbol \ for set difference, we define the observer function ϕ as follows.

Initial turn:

$$\phi(x_0, x_0, N) := \begin{cases} \chi_A & \text{if } x_0 \notin T_A \cup T_D; \\ \neg & \text{otherwise.} \end{cases}$$

During a turn of C_A that started at the state x of Σ:

$$\phi(x, z, A) := \begin{cases} (z, \chi_D) & \text{if } z \in S_A(x) \setminus T_A; \\ \neg & \text{otherwise.} \end{cases}$$

During a turn of C_D that started at the state x:

$$\phi(x, z, D) := \begin{cases} (z, \chi_A) & \text{if } z \in S_D(x) \setminus T_D; \\ \neg & \text{otherwise.} \end{cases}$$

Note that $\mathcal{O}$ generates no further action after a stable state was reached at a target state, in compliance with Rule 2.4 *(iv)*.

Part II. Building C_A^a: Assume that the configuration of Figure 1 is in a stable state either at its initial state or at the end of a C_D turn; then, Σ, C_A, and C_D are all in stable states. Let x be the stable state of Σ, where $x = x_0$ initially or, otherwise, x has been reached in compliance with Rule 2.4 at the end of a C_D turn. According to Part I of this construction, the observer $\mathcal{O}$ triggers the character χ_A upon detecting the state x of Σ; this trigger is used in the present construction to activate C_A^a. The controller C_D rests in a stable state during the action of C_A^a.

The first design step is to select a member of the local sink $x' \in S_A(x)$ with the following property: there is an A-action input string $u = u_1u_2\ldots u_{q_A(x)} \in (A \times \neg)^+$ that takes Σ from a stable state at x to a stable state at x' in fundamental mode operation and without encountering any members of $S_A(x)$ along the way. (The derivation of the string u is discussed in Sections 3–5.) Here, $q_A(x) \geq 0$ is an integer, where $q_A(x) = 0$ indicates that C_A takes no action, leaving Σ at its current state. Let $x_1x_2\ldots x_{d_A(x)}$ be the string of states through which u takes Σ; here, $x_{d_A(x)} = x'$, where $x' = x$ if $q_A(x) = 0$. Considering that, in fundamental mode operation, input triggers occur only at stable states, there are integers $i_1, i_2, \ldots, i_{q_A(x)}$, where $i_{q_A(x)} = d_A(x)$, such that

$$x_{i_1} = s(x, u_1) \text{ and } x_{i_{k+1}} = s(x_{i_k}, u_{k+1}), \quad k = 1, 2, \ldots, q_A(x) - 1, \tag{2.12}$$

where $x_{i_{q_A(x)}} = x'$.

Let Ξ_A be the state set of C_A^a. We associate with the transition chain (2.12) a subset of states, say $\{\xi^0, \xi_A^1(x), \xi_A^2(x), \ldots, \xi_A^{q_A(x)}(x)\} \subseteq \Xi_A$, where ξ^0 is the stable state of C_A^a at the start and at the end of any of its turns.

Let $\varphi_A\colon \Xi_A \times X \times \{\chi_A, \chi_D\} \to \Xi_A\colon (\xi, z, \chi) \mapsto \varphi_A(\xi, z, \chi)$ be the recursion function of C_A^a; its values are determined by the current state ξ of C_A^a, by the current state z of Σ, and by the observer signal χ. In compliance with Rule 2.4, set

$$\varphi_A(\xi, z, \chi) := \begin{cases} \xi^0 & \text{if } \xi = \xi^0 \text{ and } \chi \neq \chi_A; \\ \xi_A^1(x) & \text{if } \xi = \xi^0, z = x, \text{ and } \chi = \chi_A; \\ \xi_A^{k+1}(x) & \text{if } \xi = \xi_A^k(x), z = x_{i_k}, \text{ and } \chi = \neg, \\ & \quad k = 1, 2, \ldots, q_A(x) - 1; \\ \xi^0 & \text{if } \xi = \xi_A^{q_A(x)}(x), z = x' \text{ and } \chi = \neg. \end{cases} \tag{2.13}$$

The output function $\eta_A\colon \Xi_A \to (A \times \neg)$ of C_A^a is then

$$\eta_A(\xi) := \begin{cases} (\neg, \neg) & \text{if } \xi = \xi^0; \\ (u_k, \neg) & \text{if } \xi = \xi_A^{i_k}(x), k = 1, 2, \ldots, q_A(x). \end{cases} \tag{2.14}$$

This completes the construction of C_A^a. A brief examination shows that this construction guarantees that C_A drives Σ in fundamental mode operation in compliance with Rule 2.4 (compare to Murphy et al. (2002, 2003)).

Part III. Construction of C_D: The controller C_D consists of the observer $\mathcal{O}$ constructed in Part I and an action part C_D^a that is obtained by replacing the subscript A by the subscript D in the construction of C_A^a above. This concludes the construction of the controllers C_A and C_D.

3. Matrix representations

3.1 Strings of pairs

Consider an asynchronous trigger machine $\Sigma = (A, D, X, f, x_0)$ with two input alphabets A and D. Each command of Σ is represented by a pair of characters $(u, \upsilon) \in A \times D$. Fundamental mode operation (Rule 1.12) allows only one of these characters to trigger at a time; thus, every input pair (u, υ) satisfies either $u = \neg$ or $\upsilon = \neg$. We use the symbol $A \otimes D$ to denote all pairs of $A \times D$ in which one member is $\neg$, namely,

$$A \otimes D := \{(u, \upsilon) \in A \times D : u = \neg \text{ or } \upsilon = \neg\}. \quad (3.1)$$

As always, $(A \otimes D)^+$ represents the family of all non-empty strings of members of $A \otimes D$. For example, the string $(u_1, \neg)(u_2, \neg)(\neg, \upsilon_3)$ is a typical member of $(A \otimes D)^+$. It is convenient to use the character N – a character not included in the sets A, D, or X – to represent the empty input set. Then, *concatenation* of two members $(a, b), (a', b') \in (A \otimes D)^+ \cup N$ is defined by

$$\mathrm{conc}((a, b), (a', b')) : = \begin{cases} (aa', bb') & \text{if } (a, b), (a', b') \in (A \otimes D)^+; \\ N & \text{if } (a, b) = N \text{ or } (a', b') = N. \end{cases} \quad (3.2)$$

To work with sets of input strings, we use the following operation that is similar to union of sets. Given two subsets $S_1, S_2 \subseteq (A \otimes D)^+ \cup N$, their *sum* is

$$S_1 \uplus S_2 := \begin{cases} S_1 \cup S_2 & \text{if } S_1 \neq N \text{ and } S_2 \neq N, \\ S_1 & \text{if } S_2 = N, \\ S_2 & \text{if } S_1 = N. \end{cases}$$

Then, the *concatenation* of two sets of strings $S_1, S_2 \subseteq (A \otimes D)^+ \cup N$ is

$$\mathrm{conc}(S_1, S_2) := \biguplus_{\sigma_1 \in S_1, \sigma_2 \in S_2} \mathrm{conc}(\sigma_1, \sigma_2), \quad (3.3)$$

where σ_1 and σ_2 are individual strings.

3.2 The matrix of stable transitions

The matrix of stable transitions (Murphy et al., 2002, 2003) has proven to be an important tool in the design of controllers for asynchronous machines, and it will serve an important role in our present discussion as well. To adapt the one-step matrix of stable transitions to machines with multiple inputs, let $\Sigma = (A, D, X, Y, f, x_0)$ be an asynchronous trigger machine with inputs A and D, state set $X = \{x^1, x^2, \ldots, x^n\}$, and stable recursion function $s: X \times (A \otimes D) \to X$. Denote by $A \times \neg$ the set of all pairs of the form $(u, \neg)$, where $u \in A$, and, similarly, by $\neg \times D$ the set of all pairs of the form $(\neg, \upsilon)$, where $\upsilon \in D$. Here, the set $A \times \neg$ describes one-step A-action, while the set $\neg \times D$ describes one-step D-action .

We build for Σ two $n \times n$ one-step matrices of stable transitions – the first one describes one-step A-action and the second one describes one-step D-action; in both cases, the character N indicates impossible transitions:

$$\begin{aligned} R^1_{ij}(\Sigma, A) &:= \begin{cases} \{(u, \neg) \in (A \times \neg) : x^j = s(x^i, (u, \neg))\} & \text{if } x^j \in s(x^i, (A \times \neg)), \\ N & \text{otherwise;} \end{cases} \\ R^1_{ij}(\Sigma, D) &:= \begin{cases} \{(\neg, \upsilon) \in (\neg \times D) : x^j = s(x^i, (\neg, \upsilon))\} & \text{if } x^j \in s(x^i, (\neg \times D)), \\ N & \text{otherwise;} \end{cases} \end{aligned} \quad (3.4)$$

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

Example 3.5: Referring to the machine Σ of Example 2.3, a direct examination of Table 1 yields

$$R^1(\Sigma, A) = \begin{pmatrix} \{(\neg,\neg), (a^1,\neg)\} & N & N & \{(a^2,\neg)\} & N \\ \{(a^1,\neg)\} & \{(\neg,\neg)\} & \{(a^2,\neg)\} & N & N \\ \{(a^1,\neg), (a^2,\neg)\} & N & \{(\neg,\neg)\} & N & N \\ \{(a^2,\neg)\} & N & N & \{(\neg,\neg), (a^1,\neg)\} & N \\ \{(a^1,\neg)\} & N & N & \{(a^2,\neg)\} & \{(\neg,\neg)\} \end{pmatrix},$$

$$R^1(\Sigma, D) = \begin{pmatrix} \{(\neg,\neg), (\neg,d^1)\} & \{(\neg,d^2)\} & N & N & N \\ \{(\neg,d^2)\} & \{(\neg,\neg), (\neg,d^1)\} & N & N & N \\ N & N & \{(\neg,\neg)\} & \{(\neg,d^1)\} & \{(\neg,d^2)\} \\ \{(\neg,d^1)\} & \{(\neg,d^2)\} & N & \{(\neg,\neg)\} & N \\ \{(\neg,d^1)\} & \{(\neg,d^2)\} & N & N & \{(\neg,\neg)\} \end{pmatrix}.$$

To define operations between matrices of strings, let E and F be two $n \times n$ matrices whose entries are subsets of $(A \otimes D)^+ \cup N$. Then, the i, j entry of the *sum* $E + F$ is

$$(E + F)_{ij} := E_{ij} \uplus F_{ij},\ i, j = 1, 2, \ldots, n,$$

and the i, j entry of the product EF is

$$(EF)_{ij} := \biguplus_{\ell=1,\ldots,n} \mathrm{conc}(E_{i\ell}, F_{\ell j}),\ i, j = 1, 2, \ldots, n. \quad (3.6)$$

It is convenient here to define the $n \times n$ 'identity' matrix

$$I := \begin{pmatrix} (\neg,\neg) & N & & \cdots & N \\ N & (\neg,\neg) & N & \cdots & N \\ \vdots & & & \cdots & \\ N & & & \cdots\ N & (\neg,\neg) \end{pmatrix}.$$

Then, the powers of a matrix are defined by

$$E^0 := I,$$
$$E^\ell := EE^{\ell-1},\ \ell = 1, 2, 3, \ldots$$

The combination $E^{(p)}$ of all powers up to the power of $p \geq 1$ is

$$E^{(p)} := E^1 + E^2 + \cdots + E^p. \tag{3.7}$$

Using these operations, we build the matrices $(R^1(\Sigma, A))^{(n-1)}$ and $(R^1(\Sigma, D))^{(n-1)}$. The following statement paraphrases a statement of Murphy et al. (2002, 2003) and is proved similarly.

Proposition 3.8: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the one-step matrices of stable transitions $R^1(\Sigma, A)$ and $R^1(\Sigma, D)$. Then, the following are true for any stable states $x^i, x^j \in X$:*

(*i*) *x^j is stably reachable from x^i through A-action if and only if $(R^1(\Sigma, A))^{(n-1)}_{ij} \neq N$.*
(*ii*) *x^j is stably reachable from x^i through D-action if and only if $(R^1(\Sigma, D))^{(n-1)}_{ij} \neq N$.*

To relate the results of Proposition 3.8 to the control configuration of Figure 1, we must take into account the fact that Rule 2.4 restricts the machine Σ to the local sink. To enforce this requirement, we construct the following matrices. Recall that a string $u^1 \in A^+$ is a *strict prefix* of a string $u^2 \in A^+$ if there is a non-empty string $u^3 \in A^+$ such that $u^2 = u^1u^3$.

Construction 3.9: The matrix of A-stable transitions: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the weight function ω: $X \to Z$, and the one-step matrix of stable transitions $R^1(\Sigma, A)$.

Perform the following steps on entries of $(R^1(\Sigma, A))^{(n-1)}$ for each $i = 1, 2, \ldots, n$:

Step 1: If $x^i \in T_A \cup T_D$, then replace all off-diagonal entries of row i by N.
Step 2: If $x^i \in T_D$, then replace by N all off-diagonal entries of column i.
Step 3: Let $\zeta_A(i)$ be the set of all remaining integers $j \in \{1, 2, \ldots, n\}$ for which position j of row i is not N. If $\zeta_A(i) \neq \varnothing$, denote by $w_A(i)$ the minimal weight of a state that is stably reachable from x^i through A-action, namely,

$$w_A(i) = \min_{j \in \zeta_A(i)} \omega(x^j).$$

Replace by N all entries j of row i for which $\omega(x^j) > w_A(i)$.
Step 4: In row i, delete all strings that include as a strict prefix a string that appears anywhere else in row i. If any empty entries are obtained, replace them by the character N.
Step 5: If an entry includes the string $(\neg, \neg)$, then delete all other strings from this entry.

The resulting matrix is denoted by $R(\Sigma, A)$ and is called the *matrix of A-stable transitions* of Σ.

Example 3.10: Consider the machine Σ of Example 2.3. Raising the matrix $R^1(\Sigma, A)$ of Example 3.5 to the powers 1, 2, 3, and 4, finding $(R^1(\Sigma, A))^{(4)}$, and applying Construction 3.9, yields the matrix of A-stable transitions of Σ: (For typographical reasons, no more than one string is listed in each entry.)

$$R(\Sigma, A) = \begin{pmatrix} N & N & N & \{(a^2,\neg)\} & N \\ N & N & \{(a^2,\neg)\} & N & N \\ N & N & \{(\neg,\neg)\} & N & N \\ N & N & N & \{(\neg,\neg)\} & N \\ N & N & N & N & \{(\neg,\neg)\} \end{pmatrix}.$$

Remark 3.11: When calculating the matrix $R(\Sigma, A)$, it is usually not necessary to calculate all entries of $(R^1(\Sigma, A))^{(n-1)}$. Instead, one can first determine which entries of $R(\Sigma, A)$ are N and avoid calculating the corresponding entries of $(R^1(\Sigma, A))^{(n-1)}$.

The *A-action skeleton matrix* $K(\Sigma, A)$ of the machine Σ is defined from the matrix of A-stable transitions $R(\Sigma, A)$ by

$$K_{ij}(\Sigma, A) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma, A) \neq N, \\ 0 & \text{otherwise,} \end{cases} \quad j = 1, 2, \ldots, n. \tag{3.12}$$

Example 3.13: From Example 3.10, we obtain for the machine Σ of Example 2.3:

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

The significance of the A-action skeleton matrix stems from the next statement.

Proposition 3.14: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the matrix of A-stable transitions $R(\Sigma, A)$, and the A-action skeleton matrix $K(\Sigma, A)$. Then, under Rule 2.4, one turn of A-action can take Σ from a state x^i to a state x^j if and only if $K_{ij}(\Sigma, A) = 1$.*

When $K_{ij}(\Sigma, A) = 1$, the entry $R_{ij}(\Sigma, A)$ consists of A-action input strings that take Σ from x^i to x^j in compliance with Rule 2.4.

Proof: Considering Proposition 3.8*(i)*, Construction 3.9 of $R(\Sigma, A)$ has the following implications. Step 1 of Construction 3.9 guarantees that Σ cannot move away from a target state, once it has reached one; this fulfils Rule 2.4*(iv)*. Step 2 of Construction 3.9 assures compliance with Rule 2.4*(iii)(b)* by eliminating all input strings that may cause A-action to drive Σ into the defensive target set T_D. Finally, Steps 3–5 of Construction 3.9 assure compliance with Rule 2.4*(ii)* by allowing only first-time encounters with states of the local A-sink. The last sentence of the proposition follows by construction. This completes our proof. □

Replacing input A by input D in Construction 3.9 leads to the following.

Construction 3.15: The matrix of D-stable transitions: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the weight function ω: $X \to Z$, and the one-step matrix of stable transitions $R^1(\Sigma, D)$.

Perform the following steps on the entries of $(R^1(\Sigma, D))^{(n-1)}$ for each $i = 1, 2, \ldots, n$:

Step 1: If $x^i \in T_A \cup T_D$, then replace all off-diagonal entries of row i by N.

Step 2: If $x^i \in T_A$, then replace by N all off-diagonal entries of column i.

Step 3: Let $\zeta_D(i)$ be the set of all remaining integers $j \in \{1, 2, \ldots, n\}$ for which position j of row i is not N. If $\zeta_D(i) \neq \varnothing$, denote by $w_D(i)$ the maximal weight of a state that is stably reachable from x^i through D-action, namely,

$$w_D(i) = \max_{j \in \zeta_D(i)} \omega(x^j).$$

Replace by N all entries j of row i for which $\omega(x^j) < w_D(i)$.

Step 4: In row i, delete all input strings that include as a strict prefix a string that appears anywhere else in row i. If any empty entries are obtained, replace them by the character N.

Step 5: If an entry includes the string $(\neg, \neg)$, then remove all other strings from the entry.

The resulting matrix is denoted by $R(\Sigma, D)$ and is called the *matrix of D-stable transitions* of Σ.

Example 3.16: Consider the machine Σ of Example 2.3. Raising the matrix $R^1(\Sigma, D)$ of Example 3.5 to the powers 1, 2, 3, and 4, finding $(R^1(\Sigma, A))^{(4)}$, and applying Construction 3.15, yields the matrix of D-stable transitions of Σ: (For typographical simplicity, no more than one string is listed in each entry.)

$$R(\Sigma, D) = \begin{pmatrix} \{(\neg,\neg)\} & \{(\neg,d^2)\} & N & N & N \\ \{(\neg,d^2)\} & \{(\neg,\neg)\} & N & N & N \\ N & N & \{(\neg,\neg)\} & N & N \\ \{(\neg,d^1)\} & \{(\neg,d^2)\} & N & N & N \\ N & N & N & N & \{(\neg,\neg)\} \end{pmatrix}.$$

Remark 3.17: Remark 3.11 also applies to the construction of the matrix $R(\Sigma, D)$.

The *D-action skeleton matrix* $K(\Sigma, D)$ of the machine Σ is then

$$K_{ij}(\Sigma, D) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma, D) \neq N, \\ 0 & \text{otherwise,} \end{cases} \quad j = 1, 2, \ldots, n. \tag{3.18}$$

Example 3.19: From Example 3.16, the D-skeleton matrix of the machine Σ of Example 2.3 is:

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

The following statement is analogous to Proposition 3.14.

Proposition 3.20: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the matrix of D-stable transitions $R(\Sigma, D)$, and the D-action skeleton matrix $K(\Sigma, D)$. Then, under Rule 2.4, one turn of D-action can take Σ from a state x^i to a state x^j if and only if $K_{ij}(\Sigma, D) = 1$.*

When $K_{ij}(\Sigma, D) = 1$, the entry $R_{ij}(\Sigma, D)$ consists of D-action input strings that take Σ from x^i to x^j in compliance with Rule 2.4.

Recall from Automata theory that a *terminal state* is a state from which a machine cannot be moved. Now, if a diagonal entry of $K(\Sigma, A)$ is 1, say $K_{jj}(\Sigma, A) = 1$, then x^j is a member of the local A-sink of itself. In that case, Rule 2.4*(ii)* implies that C_A will leave Σ at x^j, and hence there will be no transitions out from x^j by A-action. It follows then by Proposition 3.20 that all off-diagonal entries of row j of $K(\Sigma, A)$ are zero. Similarly, if a diagonal entry of $K(\Sigma, D)$ is 1, then all other entries of that row are zero. These arguments lead to the following.

Proposition 3.21: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine operated in compliance with Rule 2.4 and having the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the skeleton matrices $K(\Sigma, A)$ and $K(\Sigma, D)$. Then, the following are true for any pair of integer $j \in \{1, 2, \ldots, n\}$.*

(*i*) *x^j is a terminal state for A-action if and only if $K_{jj}(\Sigma, A) = 1$.*
(*ii*) *x^j is a terminal state for D-action if and only if $K_{jj}(\Sigma, D) = 1$.*

4. Successive controller turns

4.1 The compound matrix of stable transitions

Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the matrices of stable transitions $R(\Sigma, A)$ and $R(\Sigma, D)$. Assume that Σ is controlled by the two controllers C_A and C_D of Figure 1 in compliance with Rule 2.4. The controllers C_A and C_D operate in alternate turns starting from the initial state $x_0 = x^i$ of Σ, where C_A is the first controller to act. By Proposition 3.14, the non-N entries of row i of $R(\Sigma, A)$ characterise all stable states to which C_A could take Σ at the end of its first turn. The actual state to which Σ will be taken by C_A depends on selections made during the design of C_A. According to Proposition 3.14, the (i, j) entry of $R(\Sigma, A)$ consists of strings that C_A may generate to drive Σ from the state x^i to the state x^j. To achieve a transition to x^j, one of these strings is selected by the designer of C_A and is used in Construction 2.11 as part of the implementation of C_A.

Once C_A has completed its turn, the turn of C_D starts. Using the matrix multiplication (3.6), the stable states to which C_D could take Σ at the end of its turn are characterised by the non-N entries of row i of the product

$$R(\Sigma, A)R(\Sigma, D). \tag{4.1}$$

Again here, only one of these states will actually be reached; its identity is determined by selections made during the design of C_A and C_D. Once C_A was designed, it follows by Proposition 3.20 that the non-N entries of $R(\Sigma, D)$ indicate strings that C_D may generate to drive Σ to a corresponding state at the end of its turn; one of these strings is selected by the designer of C_D toward the implementation of C_D through Construction 2.11.

The next turn is again a turn of C_A; the stable states that Σ could reach at the end of this turn are characterised by all non-N entries of row i of the product

$$R(\Sigma, A)R(\Sigma, D)R(\Sigma, A).$$

At the succeeding turn, C_D acts again, and the stable states that Σ could reach at the end of the turn are characterised by the non-N entries of row i of the product

$$R(\Sigma, A)R(\Sigma, D)R(\Sigma, A)R(\Sigma, D). \tag{4.2}$$

And so on and on. Of these states, the state actually reached by Σ is determined by selections made in the design of the two controllers C_A and C_D.

As Σ has only n states, it is possible to characterise in finite terms the states that Σ could reach at the end of any number of controller turns. To derive this characterisation, we use the following matrix.

Definition 4.3: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the matrices of stable transitions $R(\Sigma, A)$ and $R(\Sigma, D)$. The *compound matrix of stable transitions* of Σ is

$$R(\Sigma) := \sum_{i=1}^{n-1}[(R(\Sigma, A)R(\Sigma, D))^{i-1}R(\Sigma, A) + (R(\Sigma, A)R(\Sigma, D))^{i}]. \tag{4.4}$$

The *compound skeleton matrix* of Σ is

$$K_{ij}(\Sigma) := \begin{cases} 1 & \text{if } R_{ij}(\Sigma) \neq N, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Example 4.6: The compound matrix of stable transitions for the machine Σ of Example 2.3 is given by

$$\begin{aligned} R(\Sigma) = {} & R(\Sigma, A) + (R(\Sigma, A)R(\Sigma, D)) \\ & + (R(\Sigma, A)R(\Sigma, D))R(\Sigma, A) + (R(\Sigma, A)R(\Sigma, D))^2 \\ & + (R(\Sigma, A)R(\Sigma, D))^2 R(\Sigma, A) + (R(\Sigma, A)R(\Sigma, D))^3 \\ & + (R(\Sigma, A)R(\Sigma, D))^3 R(\Sigma, A) + (R(\Sigma, A)R(\Sigma, D))^4, \end{aligned}$$

where $R(\Sigma, A)$ and $R(\Sigma, D)$ are given in Examples 3.10 and 3.16.

The significance of the compound skeleton matrix is as follows.

Lemma 4.7: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$, the initial state $x_0 = x^i$, and the compound skeleton matrix $K(\Sigma)$. Assume that Σ is operated in compliance with Rule 2.4 by two controllers C_A and C_D as depicted in Figure 1. Then, the following are equivalent.*

(*i*) *There are controllers C_A and C_D for which the state x^j is the outcome of a succession of controller turns.*
(*ii*) $K_{ij}(\Sigma) \neq 0$.

Proof: Considering that $x_0 = x^i$, it follows by Proposition 3.14 and the opening discussion of Section 4.1 that all states that could be stably reached at the end of an A-action turn are characterised by the non-N entries of row i of the following matrices: $R(\Sigma, A)$ for the first A-action turn; $R(\Sigma, A)R(\Sigma, D)R(\Sigma, A)$ for the second A-action turn; and, generally, $(R(\Sigma, A)R(\Sigma, D))^{i-1}R(\Sigma, A)$ after the i-th A-action turn. Considering that Σ has only n states, every string of more than n states must include a repetition. Counting the initial state of Σ, the string x_0, $R(\Sigma, A)$, $R(\Sigma, A)R(\Sigma, D)R(\Sigma, A), \ldots, (R(\Sigma, A)R(\Sigma, D))^{i-1}R(\Sigma, A)$ yields a string of $i + 1$ stable states. Thus, for $i > n - 1$, this string of stable states must include a repeating state. Hence, proceeding in this string beyond $i = n - 1$ does not yield new states that are stably reachable at the end of an A-action turn.

Analogously, states that could be stably reached at the end of a D-action turn are characterised by the non-N entries of the matrices $(R(\Sigma, A)R(\Sigma, D))^r$, $r = 1, 2\ldots$ As before, no new states are obtained for $r > n - 1$. In view of Definition 4.3 of the compound skeleton matrix, our proof concludes. □

We can characterise now the possible outcomes of the control process of Figure 1.

Theorem 4.8: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with state set $X = \{x^1, x^2, \ldots, x^n\}$, initial state $x_0 = x^i$, and skeleton matrices $K(\Sigma, A)$, $K(\Sigma, D)$, and $K(\Sigma)$. Assume that Σ is controlled in compliance with Rule 2.4 by two controllers C_A and C_D as depicted in Figure 1. Then, the following are valid.*

(*i*) *There are controllers C_A and C_D that guide Σ to a terminal state at x^j if and only if $K_{ij}(\Sigma) = 1$, $K_{jj}(\Sigma, A) = 1$, and $K_{jj}(\Sigma, D) = 1$.*
(*ii*) *There are controllers C_A and C_D that guide Σ into an infinite cycle if and only if there are integers $p \neq q \in \{1, 2, \ldots, n\}$ such that $K_{ip}(\Sigma) = 1$, $K_{pq}(\Sigma) = 1$, and $K_{qp}(\Sigma) = 1$.*

Proof: *(i)* In order for the state x^j to be a terminal state, x^j must be stably reachable from the initial state x^i. According to Lemma 4.7, the latter is valid if and only if $K_{ij}(\Sigma) = 1$. Furthermore, in order for x^j to be a terminal state, there must be controllers C_A and C_D for which x^j is a terminal state. The statement follows then by Proposition 3.21.

(ii) By Lemma 4.7, the requirement $K_{ip}(\Sigma) = 1$ is equivalent to the existence of controllers C_A and C_D that induce a transition from the initial state x^i to the state x^p. Furthermore, by the same lemma, the two equalities $K_{pq}(\Sigma) = 1$ and $K_{qp}(\Sigma) = 1$ are equivalent to the existence of controllers that induce transitions from x^p to x^q and back from x^q to x^p. As $p \neq q$, these transitions create an infinite cycle. □

By using Theorem 4.8*(i)*, we can characterise all possible terminal states of the machine Σ in the configuration of Figure 1, as follows.

Corollary 4.9: *Under the conditions of Theorem 4.8, let $\Theta \subseteq X$ be the set of all possible terminal states of Σ in the control configuration of Figure 1, where the initial state of Σ is x^i. Then, the following are true.*

(*i*) $\Theta = \{x^j \in X$: $K_{ij}(\Sigma) = 1$, $K_{jj}(\Sigma, A) = 1$, $K_{jj}(\Sigma, D) = 1.\}$
(*ii*) *Σ reaches a terminal state for all controllers C_A and C_D if and only if $x^q \in \Theta$ whenever $K_{iq}(\Sigma) = 1$.*
(*iii*) *There are controllers C_A and C_D that lead Σ into an infinite cycle if and only if $K_{iq}(\Sigma) = 1$ for some $x^q \notin \Theta$.*

Recall that, in order for one of the controllers to prevail, the machine Σ must stably reach a target state of that controller. By Rule 2.4, stably reached target states are always terminal states of Σ in the control configuration of Figure 1. Combining this with Corollary 4.9 yields the next statement.

Corollary 4.10: *Under the conditions of Theorem 4.8, the following are valid.*

(*i*) *There are controllers C_A and C_D for which C_A prevails if and only if $K_{ij}(\Sigma) = 1$ for some $j \in T_A$.*
(*ii*) *Any controller C_A prevails for any C_D if and only if $j \in T_A$ whenever $K_{ij}(\Sigma) = 1$.*
(*iii*) *There are controllers C_A and C_D for which C_D prevails if and only if $K_{ij}(\Sigma) = 1$ for some $j \in T_D$.*
(*iv*) *Any controller C_D prevails for any C_A if and only if $j \in T_D$ whenever $K_{ij}(\Sigma) = 1$.*

As can be seen, the compound skeleton matrix $K(\Sigma)$ characterises the potential performance of the closed-loop machine of Figure 1. Note, however, that $K(\Sigma)$ provides information only about potential outcomes of the control process; it does not provide the information necessary for designing controllers C_A and C_D that achieve a particular outcome. The design of appropriate controllers C_A and C_D requires the corresponding matrices of stable transitions $R(\Sigma, A)$ and $R(\Sigma, D)$, since these are the matrices that furnish the input strings that C_A and C_D must generate as input to Σ in order to achieve a particular outcome. Still, if one is interested preliminarily only in examining the possible outcomes of the control process of Figure 1, then the skeleton matrix is the preferred tool to use. In the next subsection, we discuss a simple technique for deriving the compound skeleton matrix $K(\Sigma)$.

4.2 *A simple computation of compound skeleton matrices*

We start by reviewing briefly the operations on skeleton matrices introduced by Murphy et al. (2002, 2003). Let K and K' be two $n \times n$ skeleton matrices, namely, two matrices of zeros and ones. The sum $K + K'$ is also an $n \times n$ skeleton matrix, and it has the entries

$$(K + K')_{ij} := \max\{K_{ij}, K'_{ij}\},\ i, j = 1, 2, \ldots, n. \tag{4.11}$$

The product KK' is again an $n \times n$ skeleton matrix, and its entries are

$$(KK')_{ij} := \max_{q=1,2,\ldots,n}\{K_{iq}K'_{qj}\},\ i, j = 1, 2, \ldots, n. \tag{4.12}$$

Under this product, powers of an $n \times n$ skeleton matrix K are given by

$$K^0 := I,$$
$$K^p := KK^{p-1},\ p = 1, 2, \ldots,$$

where I is the identity matrix of zeros and ones. Finally, for an integer $p > 0$, the *cumulative power* $K^{(p)}$ of a skeleton matrix K is

$$K^{(p)} = K + K^2 + \cdots + K^p.$$

Now, let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, x^2, \ldots, x^n\}$. Using the two skeleton matrices $K(\Sigma, A)$ and $K(\Sigma, D)$ of (3.12) and (3.18), it can be readily seen that the compound skeleton matrix of Definition 4.3 is equal to

$$K(\Sigma) = \sum_{i=1}^{n-1}\{(K(\Sigma, A)K(\Sigma, D))^{i-1}K(\Sigma, A) + (K(\Sigma, A)K(\Sigma, D))^{i}\} \tag{4.13}$$

The calculation of the compound skeleton matrix through this formula is somewhat simpler than its derivation from the compound matrix of stable transitions in (4.5), since the present calculation does not require the derivation of the compound matrix of stable transitions $R(\Sigma)$. The matrix $K(\Sigma)$ can be utilised in Theorem 4.8 to determine possible outcomes of the control process of Figure 1. We emphasise again that the matrices of stable transitions are needed for implementing appropriate controllers, since these matrices provide the strings used in Construction 2.11 to build the controllers.

5. States of certainty

In this section, we examine conditions under which the outcome of the control process of Figure 1 becomes pre-determined. Specifically, we characterise states of the controlled machine Σ from which one controller can always prevail, irrespective of actions taken by the other controller (if it employs an appropriate control strategy). We also consider the dual problem of characterising all states of Σ from which one controller can always block the other controller from prevailing, by using an appropriate control strategy. First, some terminology.

Definition 5.1: Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine operated by the two controllers C_A and C_D of Figure 1 in compliance with Rule 2.4. The controller C_A can *prevail with certainty* from a state x of Σ if C_A can always prevail after starting a turn at x, irrespective of actions taken by C_D. Similarly, C_D can *prevail with certainty* from a state x of Σ if C_D can always prevail after starting a turn at x, irrespective of actions taken by C_A.

States from which one of the controllers can prevail with certainty are states of the controlled machine Σ from which the outcome of the control process of Figure 1 is pre-ordained, irrespective of actions taken by the opposing controller. This assumes, of course, that the controller that can prevail with certainty employs an appropriate control strategy; the point is that such a strategy always exists for a controller that can prevail with certainty.

To characterise states from which a controller can prevail with certainty, we use the fact that, when a controller can prevail with certainty, the end of the control process is known – the controlled machine Σ reaches a target state of the prevailing controller. Using this fact, we proceed in a step-by-step manner backwards, characterising all states along the way from which the controller can prevail with certainty. Let's examine first the case of the controller C_A. Clearly, in order for C_A to prevail in the current turn, it must drive Σ to a member of the target set T_A in this turn. Thus, C_A must have started its turn at a state of Σ from which a member of T_A is stably reachable in one turn of A-action. Going back one turn, this means that the previous turn – a D-action turn – must have started at a state of Σ from which all states that can be stably reached in one turn of D-action are states from which C_A can stably reach a member of T_A in one turn; and so forth, going back controller turn by controller turn. To describe this process formally, we introduce some notation.

5.1 *Vector representations of states*

Consider the asynchronous trigger machine $\Sigma = (A, D, X, f, x_0)$ with the state set $X = \{x^1, x^2, \ldots, x^n\}$ and the target sets T_A and T_D. Let $\{0, 1\}^n$ be the set of all n-dimensional column vectors with entries of 0 or 1, and let $\chi: X \to \{0, 1\}^n$ be the function that assigns to a state $x^j \in X$ the column vector $\chi(x^j) = (0, 0, \ldots, 0, 1, 0, \ldots, 0)^T$, where 1 appears in position j and zeros everywhere else, and where T denotes the transpose. More generally, a set of states

$S = \{x^{i_1}, x^{i_2}, \ldots, x^{i_q}\} \subseteq X$ is represented by the column vector $\chi(S) \in \{0, 1\}^n$ that has entries of 1 in positions $i_1, i_2, \ldots, i_q$ and zeros everywhere else; the zero vector corresponds to the empty set of states. This turns χ into a set isomorphism. In our ensuing discussion, it will be convenient to make no distinction between the set S and the vector $\chi(S)$, in line with Notation 1.5*(iii)*.

Given a vector $t \in \{0, 1\}^n$ and an $n \times n$ skeleton matrix K, we define a product analogous to Equation (4.12) to yield a column vector $t' = (t'_1, t'_2, \ldots, t'_n)^T \in \{0, 1\}^n$, where

$$t'_i := \max_{q=1,2,\ldots,n} \{K_{iq} t_q\},\ i = 1, 2, \ldots, n.$$

The addition of two vectors $\vartheta = (\vartheta_1, \ldots, \vartheta_n)^T$, $\vartheta' = (\vartheta'_1, \ldots, \vartheta'_n)^T \in \{0, 1\}^n$ is defined in analogy to (4.11) by

$$(\vartheta + \vartheta')_i := \max\{\vartheta_i, \vartheta'_i\},\ i = 1, 2, \ldots, n. \tag{5.2}$$

Now, let $S(j) \subseteq X$ be the set of all states of Σ from which the state x^j is stably reachable in one turn of A-action. By the definition of the A-skeleton matrix $K(\Sigma, A)$, it follows that $S(j)$ is characterised by all entries of 1 in column j of $K(\Sigma, A)$; in other words, $\chi(S(j)) = K(\Sigma, A)\chi(x^j)$. A slight reflection shows that the last expression can be generalised into the following.

Proposition 5.3: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with the state set $X = \{x^1, \ldots, x^n\}$, the A-skeleton matrix $K(\Sigma, A)$, and the D-skeleton matrix $K(\Sigma, D)$. Then, the following are true for a set of states $S \subseteq X$:*

(*i*) *The set of all states of Σ from which a member of S is stably reachable in one turn of A-action is characterised by the vector $K(\Sigma, A)\chi(S)$.*
(*ii*) *The set of all states of Σ from which a member of S is stably reachable in one turn of D-action is characterised by the vector $K(\Sigma, D)\chi(S)$.*

Considering the target set T_A, it follows by Proposition 5.3 that the set of all states of Σ from which C_A can prevail in one turn is characterised by the column vector

$$v_A^1 := K(\Sigma, A)\chi(T_A).$$

Next, we introduce an operation of complementation: the *complement* v^c of a column vector $v \in \{0, 1\}^n$ is obtained by replacing in v every 0 by 1 and every 1 by 0. Then, the vector $(v_A^1)^c$ characterises the set of all states of Σ from which a member of T_A is not stably reachable in one turn of A-action. Consequently, the vector

$$\theta_A^1 := K(\Sigma, D)(v_A^1)^c$$

characterises all states of Σ from which one turn of D-action can prevent Σ from entering the set v_A^1. Thus, θ_A^1 characterises the set of all states of Σ from which the defensive controller C_D can prevent the adversarial controller C_A from prevailing in one subsequent turn. The complement

$$v_A^2 := (\theta_A^1)^c = (K(\Sigma, D)(v_A^1)^c)^c$$

therefore characterises all states of Σ from which D-action cannot block the controller C_A from prevailing in one subsequent turn. In other words, if, at the start of a D-turn, Σ is in a state that corresponds to an entry of 1 in v_A^2, then C_A can ultimately prevail, irrespective of actions taken by C_D.

Continuing in this manner, we progress backward from the target set T_A to characterise all states from which the controller C_A can ultimately prevail, irrespective of D-action:

$$\begin{aligned} v_A^0 &:= \chi(T_A) \\ v_A^1 &:= K(\Sigma, A)\chi(T_A) \\ v_A^i &:= \left[K(\Sigma, D)(v_A^{i-1})^c\right]^c \text{ for } i = 2, 4, 6, \ldots \\ v_A^i &:= K(\Sigma, A)v_A^{i-1} \text{ for } i = 3, 5, 7, \ldots \end{aligned}$$

Finally, using the addition operation (5.2), define the vector

$$v_A := \sum_{i=0}^{n-1} v_A^i. \tag{5.4}$$

Then, the following is true.

Theorem 5.5: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with state set $X = \{x^1, x^2, \ldots, x^n\}$ and skeleton matrices $K(\Sigma, A)$ and $K(\Sigma, D)$. Assume that Σ is operated by two controllers C_A and C_D according to the configuration of Figure 1 and in compliance with Rule 2.4. Then, the set of all states of Σ from which C_A can prevail with certainty is characterised by the vector v_A of (5.4).*

Before proving Theorem 5.5, we provide an example.

Example 5.6: Referring to the machine Σ of Example 2.3 and to the results of Examples 3.13 and 3.19, we obtain

$$\chi(T_A) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ v_A^1 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ v_A^2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$v_A^3 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ v_A^4 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

This yields

$$v_A = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

which means that, in this case, the adversarial controller prevails with certainty from the states x^2 and x^3.

Proof (of Theorem 5.5)**:** We characterise all initial states x_0 of Σ from which C_A can prevail with certainty. Now, if $x_0 \in T_A$, then C_A prevails with no controller action. Otherwise, by Rule 2.4, when C_A prevails, it is the last one to act. By the discussion leading to (5.4), the vector v_A characterises all states from which C_A can prevail with certainty within zero to $n-1$ controller turns. Thus, it remains only to show that, if C_A can prevail with certainty in $m \geq 0$ controller turns, then it can also prevail with certainty within $n-1$ or fewer controller turns.

To this end, assume that, for a state $x_0 \in X$, the controller C_A can prevail with certainty in $m \geq 0$ controller turns, where $m > n-1$. Let $x_0, x_1, x_2, \ldots, x_m$ be the states reached by Σ at the end of controller turns 0, 1, 2, ..., m, respectively, on its way to a member $x_m \in T_A$. As this string includes $m+1 > n$ states, while Σ has only n states, there must be repeating states in the string, say the states $x_{i_1} = x_{i_2}$, $i_2 > i_1$. The string of states has then the form $x_0, x_1, \ldots, x_{i_1-1}, x_{i_1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2}, x_{i_2+1}, \ldots, x_m$. Note that x_{i_1} and x_{i_2} must be the outcome of a turn by the same controller, since otherwise x_{i_2} is a start of a turn of the same controller whose turn ended at $x_{i_1} = x_{i_2}$ and hence, according to Rule 2.4, that controller would have continued to x_{i_2+1} without stopping at $x_{i_1} = x_{i_2}$. Thus, we can delete the segment $x_{i_1}, x_{i_1+1}, \ldots, x_{i_2-1}$ to obtain the shorter string $x, x_1, \ldots, x_{i_1-1}, x_{i_2}, x_{i_2+1}, \ldots, x_m$ that represents $m - (i_2 - i_1)$ controller turns. Repeating this process, we can shorten the string to $n-1$ or fewer controller turns, and our proof concludes. □

In order for the controller C_A to prevail under the conditions of Theorem 5.5, it is necessary for C_A to employ an appropriate control strategy. The theorem states that such a control strategy does exist whenever the controlled machine Σ finds itself in a state of v_A at the start of a C_A turn. The strings that C_A must generate as input to Σ in response to actions by C_D are given in the appropriate entries of the matrix of A-stable transitions $R(\Sigma, A)$. These strings are then used in Construction 2.11 to build C_A.

Obviously, from a defensive perspective, it is critical for C_D to avoid taking Σ to a state from which C_A can prevail with certainty. Thus, C_D must endeavour to constrain Σ to states characterised by the complement $(v_A)^c$. A slight reflection shows that this is possible if and only if the initial state x_0 of Σ is not itself a member of v_A, as the following statement indicates. Blocking C_A from prevailing would, of course, require appropriate design of C_D. The point of the next statement is that such design is possible under the listed conditions.

Corollary 5.7: *Under the conditions and notation of Theorem 5.5, the following two statements are equivalent.*

(*i*) *C_D can block C_A from prevailing.*
(*ii*) *The initial condition x_0 of Σ corresponds to an entry of 0 in v_A.*

Completely analogous results can be obtained for the defensive controller C_D by interchanging A and D in Theorem 5.5 and Corollary 5.7. To this end, define the quantities

$$\begin{aligned} v_D^0 &:= \chi(T_D) \\ v_D^1 &:= K(\Sigma, D)\chi(T_D) \\ v_D^i &:= (K(\Sigma, A)(v_D^{i-1})^c)^c \text{ for } i = 2, 4, 6, \ldots \\ v_D^i &:= K(\Sigma, D)v_D^{i-1} \text{ for } i = 3, 5, 7, \ldots \end{aligned}$$

$$v_D := \sum_{i=0}^{n-1} v_D^i. \tag{5.8}$$

Theorem 5.9: *Let $\Sigma = (A, D, X, f, x_0)$ be an asynchronous trigger machine with state set $X = \{x^1, x^2, \ldots, x^n\}$ and skeleton matrices $K(\Sigma, A)$ and $K(\Sigma, D)$. Assume that Σ is operated in compliance with Rule 2.4 by two controllers C_A and C_D, as depicted in Figure 1. Then, the set of all states of Σ from which C_D can prevail with certainty is characterised by the vector v_D of (5.8).*

Once C_D finds itself at the beginning of its turn in a state that corresponds to an entry of 1 in v_D, no action by C_A can prevent C_D from prevailing, if C_D is properly designed.

6. Conclusion

In this paper, we applied control theoretic techniques toward the resolution of a common predicament of modern technology – the omnipresence of adversarial software agents that are pre-programmed to harm critical computing systems and the infrastructure they manage. We concentrated on asynchronous sequential machines, since the prevalence of such machines is increasing quickly, as part of the quest for higher computing speed.

A pre-programmed adversarial agent can be viewed as an automatic controller acting on an underlying machine Σ, in an attempt to take Σ into a set of 'harmful' states – the adversarial target set T_A. To counteract the actions of the controller C_A, a defensive automatic controller C_D is connected to Σ, with the objective of taking Σ into a 'beneficial' set of states – the defensive target set T_D. The resulting control configuration is depicted in Figure 1.

As one might expect, the ability to counteract action of the adversarial controller depends on certain reachability properties of the protected machine Σ. These reachability properties can be most conveniently expressed in terms of the two skeleton matrices: the A-action skeleton matrix $K(\Sigma, A)$, which describes the stable transitions the adversarial controller can induce in Σ; and the D-action skeleton matrix $K(\Sigma, D)$, which describes the stable transitions the defensive controller can induce in Σ. Being matrices of zeros and ones, these matrices are easy to manipulate; they form effective tools for examining the possible outcome of the control process. In particular, they allow us to characterise states of certainty, namely, states from which the outcome of the control process is preordained (Theorems 5.5 and 5.9).

Once the possible outcomes of the control process have been determined, the next task is to design controllers that implement a desirable outcome. As the controllers work by applying input strings to the underlying machine Σ, we need tools that reveal the appropriate input strings. Such tools are provided by the matrices of stable transitions $R(\Sigma, A)$ and $R(\Sigma, D)$. Entries of these matrices are used by the designer of the adversarial controller C_A to determine the input strings that C_A must produce in order to achieve its adversarial objectives. Similarly, the designer of the defensive controller C_D uses entries of these matrices to determine the input strings that C_D must apply to Σ in order to achieve its defensive objectives. These strings are then used in Construction 2.11 to implement the two controllers.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Barrett, G., & Lafortune, S. (1998). Bisimulation, the supervisory control problem, and strong model matching for finite state machines. *Discrete Event Systems: Theory and Applications, 8*(4), 377–429.

Di Benedetto, M.D., Sangiovanni-Vincentelli, A., & Villa, T. (2001). Model matching for finite-state machines. *IEEE Transactions on Automatic Control, 46*(11), 1726–1743.

Geng, X., & Hammer, J. (2005). Input/output control of asynchronous sequential machines. *IEEE Transactions on Automatic Control, 50*(12), 1956–1970.

Hammer, J. (1994, December). *On some control problems in molecular biology*. *Proceedings of the IEEE conference on decision and control*, Lake Buena Vista, FL (pp. 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, NY: McGraw-Hill.

Kumar, R., Nelvagal, S., & Marcus, S.I. (1997). A discrete event systems approach for protocol conversion. *Discrete Event Systems: Theory and Applications, 7*(3), 295–315.

Martin, A.J., & Nyström, M. (2006). Asynchronous techniques for system-on-chip design. *Proceedings of IEEE, 94*(6), 1089–1120.

Murphy, T.E., Geng, X., & Hammer, J. (2002, July). *Controlling races in asynchronous sequential machines*. *Proceeding of the IFAC World congress,* Barcelona.

Murphy, T.E., Geng, X., & Hammer, J. (2003). On the control of asynchronous machines with races. *IEEE Transactions on Automatic Control, 48*(6), 1073–1081.

Peng, J., & Hammer, J. (2010). Input/output control of asynchronous sequential machines with races. *International Journal of Control, 83*(1), 125–144.

Peng, J., & Hammer, J. (2012). Bursts and output feedback control of non-deterministic asynchronous sequential machines. *European Journal of Control, 18*(3), 286–300.

Sparsø, J., & Furber, S. (2001). *Principles of asynchronous circuit design – a systems perspective*. Dordrecht: Kluwer Academic.

Szor, P. (2005). *The art of computer virus research and defense*. New Jersy, NJ: Addison-Wesley.

Thistle, J.G., & Wonham, W.M. (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*. San Francisco, CA: Morgan & Claypool.

Venkatraman, N., & Hammer, J. (2006a). On the control of asynchronous sequential machines with infinite cycles. *International Journal of Control, 79*(7), 764–785.

Venkatraman, N., & Hammer, J. (2006b). Stable realizations of asynchronous sequential machines with infinite cycles. Proceedings of 2006 Asian control conference, Bali, Indonesia (pp. 45–51).

Yang, J.–M. (2011). Model matching inclusion for input/state asynchronous sequential machines. *Automatica, 47*(3), 597–602.

Yang, J.-M., & Hammer, J. (2008). State feedback control of asynchronous sequential machines with adversarial inputs. *International Journal of Control, 81*(12), 1910–1929.

Yang, J.-M., & Hammer, J. (2010). Asynchronous sequential machines with adversarial intervention: The use of bursts. *International Journal of Control, 83*(5), 956–969.

Yang, J.-M., & Kwak, S.W. (2010). Realizing fault-tolerant asynchronous sequential machines using corrective control. *IEEE Transactions on Control Systems Technology, 18*(6), 1457–1463.

Yevtushenko, N., Villa, T., Brayton, R.K., Petrenko, A., & Sangiovanni-Vincentelli, A.L. (2008). Compositionally progressive solutions of synchronous FSM equations. *Discrete Event Systems: Theory and Applications, 18*(4), 51–89.