SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

October 2024, Vol. 67, Iss. 10, 202208:1–202208:11
https://doi.org/10.1007/s11432-022-4124-7

An STP look at logical blocking of finite state machines: formulation, detection, and search

Yongyi YAN[1*], Penglei HAO[1], Jumei YUE[2], Haitao LI[3] & Jun-E FENG[4]

[1]*College of Information Engineering, Henan University of Science and Technology, Luoyang* 471023*, China;*
[2]*College of Agricultural Equipment Engineering, Henan University of Science and Technology, Luoyang* 471023*, China;*
[3]*School of Mathematics and Statistics, Shandong Normal University, Jinan* 250014*, China;*
[4]*School of Mathematics, Shandong University, Jinan* 250100*, China*

Received 22 October 2022/Revised 10 October 2023/Accepted 10 January 2024/Published online 13 September 2024

Abstract The logical blocking of finite state machines (FSMs) is examined at the three levels of formulation, detection, and search from an STP viewpoint (semi-tensor product of matrices). The research idea regards an FSM as a logical system. The realizing method treats the event sequence exciting an FSM as the input signal of a logical system and treats the current states of an FSM as the states of a logical system. Based on a recently developed bilinear dynamic model of FSMs, a difference equation-like model is first proposed to describe the logical blocking. By defining a loop structure of FSMs and using the difference equation-like model, a criterion is built by which whether a given FSM is blocking can be easily judged. If it is, several algorithms are designed to find all the logical blocking of the FSM. Further, these results are extended to apply to the case of nondeterministic FSMs and, thus, to networks of FSMs. The proposed STP approach may provide a new angle for considering the problems of FSMs, and the presented results may strengthen the links between systems governed by human-designed rules and systems governed by natural laws.

Keywords semi-tensor product of matrices, STP, matrix approach, finite-valued systems, logical systems, logical networks, finite state machines

1 Introduction

Logical blocking of finite state machines (FSMs) often leads to the deterioration of system liveness and becomes an important and difficult task that must be addressed to be avoided in practical systems, such as resource conflicts, synchronization issues, and error handling [1]. In particular, in the field of control theory, modeling and detecting logical blocking are two major problems that need to be addressed first so that practical logical systems can be studied.

From the perspective of the (semi-tensor product of matrices) STP method, this paper considers the logical blocking of FSMs at three levels: formulation, detection, and search. Logical blocking refers to an operating condition while performing a particular task. There are usually two types of logical blocking. In the first type, referred to as a deadlock, no more events can be executed, as if an FSM is locked in a state. The second type is called a live-lock, which comprises multiple states. When an FSM encounters a livelock, even though the states within the livelock may be reachable from each other, the FSM cannot escape from these states, as if falling into a trap formed by a set of states.

The literature on the logical blocking of FSMs mostly relies on traditional methods such as table lookup, state transition diagrams, and discrete function methods [2] or designing of observers [3]. These technologies have the advantages of user-friendliness, high efficiency, and easy implementation. However, they are short of abilities to mathematically investigate the logical blocking, including live-lock and deadlock, similar to how linear and nonlinear systems are handled in the control area. Recently, scholars in the control area have shown great interest in FSMs, such as modeling, analysis, synthesis, and optimization [4]. Particularly, using the STP as the modeling tool, the state evolution of FSMs is formulated as a bilinear dynamic equation [5], which lays a stepping-stone toward studying FSMs by using control

* Corresponding author (email: yyyan@mail.nankai.edu.cn)

theory. Therefore, there are sufficient reasons to believe that the STP method has unique advantages in exploring the logical blocking of FSMs [6].

Inspired by the above, namely, the limitations of the traditional methods in studying the evolution of FSMs and the inherent advantages of the STP approach in modeling logical systems, this paper considers the formulation, detection, and search of logical blocking of FSMs from the viewpoint of control theory. The aim is to establish a framework for formulating, detecting, and searching logical blocking. In this paper, we view an FSM as a logical dynamic system and follow the way of handling linear and nonlinear systems in control theory; the event strings exciting an FSM are treated as an input to a system, and the current state of the machine is simply considered the system state. According to the newly proposed bilinear model of FSMs [5], we first establish a difference equation-like model, which allows us to examine FSMs by handling the equation-like linear systems. Then, we introduce the concept of loop structure by generalizing the deadlock and live-lock to more generally describe the logical blocking and propose an algebraic criterion that can easily judge whether a given FSM is logical blocking. Using this criterion, several algorithms are designed to search all the loops of an FSM with loops. The opportunity of finding the loops gives us a chance to improve or remove the blocking from an FSM and further enhance the liveness. With slight modifications, these findings also apply to FSM networks.

To the authors' knowledge, the STP approach has not been reported for formulating, detecting, and searching logical blocking of FSMs or for analyzing and synthesizing logical blocking. The presented results obtained by the STP approach strengthen the connections between systems governed by human-designed rules and systems governed by natural laws.

2 Preliminaries: FSMs and STP of matrices

2.1 Finite state machines (FSMs)

Definition 1 ([7]). An FSM M is a 5-tuple $M = (X, E, f, x_0, X_m)$, where the components $X = \{x_1, x_2, ..., x_n\}$, $E = \{e_1, e_2, ..., e_m\}$, f, x_0 and $X_m \subset X$ are the state set, event set, transition function, initial state and the set of accepting states, respectively.

FSM M is deterministic if f is the map $X \times E \to X$; otherwise, i.e., f is the map $X \times E \to 2^X$ ($2^X$is the power set of X), it is nondeterministic. The expression $f(x_i, e_j) = \{x_{k1}, x_{k2}, ..., x_{kl}\}$ means a transition that is triggered by the event e_j from the state x_i to the states $x_{k1}, x_{k2}, ..., x_{kl}$.

FSM M works as follows. M starts at the initial state x_0 and transforms to the state $f(x_0, e_i) \in X$ if the event $e_i \in E$ is available at x_0. Next, if the event $e_j \in E$ is available at the new state $f(x_0, e_i)$, then M shifts to the state $f(f(x_0, e_i), e_j)$. This dynamic behavior continues according to the scheme defined by the transition function f.

An accepting state $x \in X_m$ is used to indicate whether an event string having been read by M is acceptable by M. If the event string shifts M to x from x_0, the string is considered acceptable by M; otherwise, unacceptable by M.

FSM M is said to be blocking if M reaches an unaccepting state or enters a group of unaccepting states and cannot exit them. Such blocking dynamics will be discussed and formulated by "loop structures" in Section 4.

2.2 The STP of matrices

Definition 2 ([8]). The STP of two matrices of any sizes $M \in \mathbf{M}_{m\times n}$ and $N \in \mathbf{M}_{p\times q}$, denoted by $M \ltimes N$, is defined by $M \ltimes N = (M \otimes I_{s/n})(N \otimes I_{s/p})$, where s=lcm(n, p), lcm$(\cdot, \cdot)$ is the least common multiple function, and $\otimes$ denotes the Kronecker product of matrices.

This new technique not only has special properties that traditional products of matrices do not have (as described below) but also conquers an inherent faultiness of traditional products, such as incomplete compatibility, and a detailed description can be found in [8]. This technology has been successfully applied in engineering and scientific fields and has achieved outstanding results, such as logic network [9, 10], networked game network [11, 12], finite state automation [13, 14], Boolean network [15, 16], and optimal control [17, 18].

Proposition 1 ([19]). Let $A \in \mathbf{M}_{m\times n}$, $B \in \mathbf{M}_{p\times q}$, and $C \in \mathbf{M}_{r\times s}$ be three matrices of any size; then, $(A \ltimes B) \ltimes C = A \ltimes (B \ltimes C)$.

Proposition 2 ([20]). For column vectors X and Y that have m and n dimensions, respectively, we have $W_{[m,n]} \ltimes X \ltimes Y = Y \ltimes X$ and $W_{[n,m]} \ltimes Y \ltimes X = X \ltimes Y$, where $W_{[m,n]}$ is defined in [20].

Remark 1. Since the traditional product of matrices is a special case of the STP (when $n = p$ in Definition 2, the STP is equivalent to the former), the product of matrices in this paper defaults to the STP and the STP symbol "$\ltimes$" is omitted except for special emphasis.

3 Difference equation-like model of DFSMs

This section aims to formulate the dynamics of deterministic finite state machines (DFSMs) by following a similar approach in the field of control theory that is extensively used to model, analyze, and control dynamic systems. Based on the bilinear dynamic equations of DFSMs developed in [5], a difference equation-like model is proposed. Then, using this model, the logical blocking of FSMs is investigated anew by being described as sample paths that are generated by the proposed model. We start by representing FSMs as vectorization forms.

Vectorization of an FSM. For an FSM $M = (X, E, f, x_0, X_m)$, the state set and event set are $X = \{x_1, x_2, ..., x_n\}$ and $E = \{e_1, e_2, ..., e_m\}$, respectively. Use the vector δ_n^i to identify the state $x_i \in X$ and δ_n^i is referred as to the vectorization of x_i, $1 \leqslant i \leqslant n$. Thus the state set X can be represented by $\Delta_n = \{\delta_n^1, \delta_n^2, ..., \delta_n^n\}$. Similarly, use δ_m^j to identify event $e_j \in E$ and δ_m^j is referred as to the vectorization of e_j,$1 \leqslant j \leqslant m$. E can thus be represented by $\Delta_m = \{\delta_m^1, \delta_m^2, ..., \delta_m^m\}$. Consequently, the transition function can be represented by $f(\delta_n^i, \delta_m^j) = \delta_n^k$. Both types of representations are used as appropriate for ease in this paper. Generally, the vectorization of FSMs is used for analytic derivation and the original version is used for synthetic description.

3.1 Difference equation-like model of DFSMs

The following bilinear dynamic equation (BDE) underlies the difference equation-like model of DFSMs proposed in this paper.

Proposition 3 ([14, 21]). The dynamics of a DFSM M triggered by the event string $u(t) = e_1 e_2 \cdots e_t$ can be formulated as the following BDE: $x(t+1) = F^t x_0 u(t)$, in which $x(t)$ denotes the state at the logical time t; x_0 denotes initial state; $u(t)$ represents the event string of length t from the initial time to the current time t; $x(t)$, x_0, and $u(t)$ are their vectorization, specifically, $u(t) = \ltimes_{i=1}^t \delta_m^i$; and F is the structure matrix of M defined as $F = GW_{[n,m]}$, where $G_{i(s,t)} = 1$ if $f(x_t, e_i) = x_s$, and $G_{i(s,t)} = 0$, otherwise.

Proposition 3 shows that if the input event sequence $u(t)$ is known, the BDE model can predict the dynamics in the future t time step at the initial state x_0. This attribute allows us to study the dynamics between any logical times t_1 and t_2, $t_1 < t_2$, which is described as the following Theorem 1, i.e., the difference equation-like formulation of DFSMs.

Theorem 1. Assume that an FSM M is motivated by an event string $e_{t_1} e_{t_1+1} \cdots e_{t_2}$ at logical time t_1. Then, the dynamics of M between logical times t_1 and t_2 can be depicted by

$$x(t_2) = F^{t_2 - t_1} x(t_1) i(t_1 + 2) \cdots i(t_2). \tag{1}$$

For the special case of two successive times t and $t+1$, the dynamic behavior of M is

$$x(t+1) = F x(t) i(t+1), \tag{2}$$

where $i(t)$ is the input event at logical time t.

Proof. Eq. (1) follows from a direct observation of Proposition 3. In fact, Proposition 3 implies that if M receives the event string at initial state x_0, then the states at time t_1 and t_2 are $x(t_1) = F^{t_1} x_0 u(t_1)$ and $x(t_2) = F^{t_2} x_0 u(t_2)$, respectively.

Using Proposition 1, it follows that $x(t_2) = F^{t_2 - t_1} F^{t_1} x_0 u(t_2) = F^{t_2 - t_1} F^{t_1} x_0 u(t_1) i(t_1 + 1) \cdots i(t_2) = F^{t_2 - t_1} \left(F^{t_1} x_0 u(t_1)\right) i(t_1 + 1) \cdots i(t_2) = F^{t_2 - t_1} x(t_1) i(t_1 + 1) \cdots i(t_2)$.

Eq. (1) is then proved. Eq. (2) follows from (1) by taking $t_2 = t + 1$ and $t_1 = t$. This result completes the proof of Theorem 1.

Remark 2. Interestingly, although Eqs. (1) and (2) resemble the difference equations of the control theory, they are not. Nonetheless, they give the state transitions of two or more consecutive logical

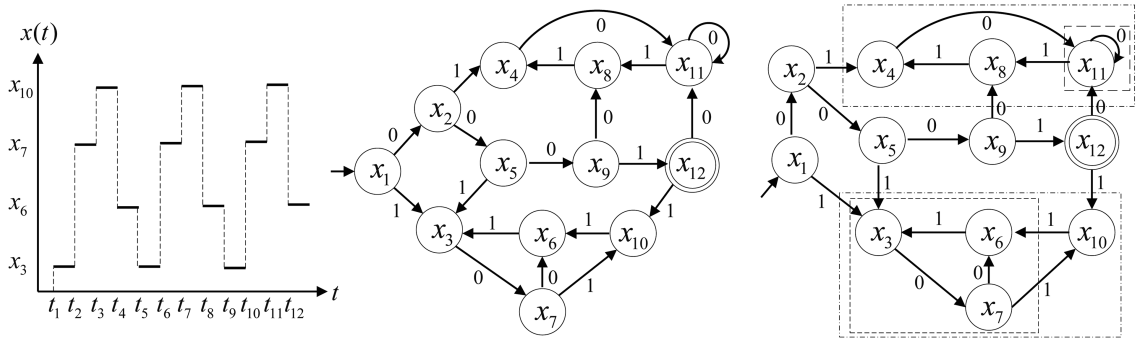

Figure 1 A sample path of the FSM and illustrations of Algorithms 2 and 3.

times. Therefore, Eqs. (1) and (2) enable us to further study the dynamic characteristics of DFSMs by using the trajectory curve. The fundamental advantage of using the STP method to study FSMs is the ability to describe the dynamic evolution of FSMs as bilinear dynamic equations, which allows for the study of FSMs within the control theory framework. This attribute is beyond the capabilities of traditional methods, which mainly include table lookup, state transition diagram, and discrete function approaches. These methods cannot model the dynamics of FSMs because the evolution of FSMs is non-time-driven, and the states undergo discrete jumps, which cannot be described with differential equations and difference equations.

3.2 State trajectory of DFSMs

With the preparation of the previous section, that is, the established difference equation-like formulation, in this section, we first draw the state trajectory of FSMs to investigate the dynamic behaviors, which provides an intuitive approach to studying the logical blocking of FSMs.

Inspired by the method of drawing a motion track of a dynamic system in control theory, we choose a given event string and an initial state, such as the string $u(t) = e_{i_1}e_{i_2}\cdots e_{i_t}$ and the initial state x_0 and time $t = 0$. According to Eq. (2), we can then obtain a series of states associated with the event string. Using the state sequence, we can draw the state evolution track of an FSM. Taking the FSM shown in Example 1 as an example, let the initial state be $x_0 = \delta_{12}^3$ and the event string be $u(t) = \delta_2^1\delta_2^2\delta_2^2\delta_2^2\delta_2^1\delta_2^2\delta_2^2\delta_2^2\delta_2^1\delta_2^2\delta_2^2\delta_2^2$.

Substituting $x_0 = \delta_{12}^3$ and the following states $i(1) = \delta_2^1, i(2) = \delta_2^2, i(3) = \delta_2^2, i(4) = \delta_2^1, i(5) = \delta_2^2, i(6) = \delta_2^2, i(7) = \delta_2^1, i(8) = \delta_2^2, i(9) = \delta_2^2$ into Eq. (2) gives the following consecutive set of states: $\delta_{12}^3, \delta_{12}^7, \delta_{12}^{10}, \delta_{12}^6, \delta_{12}^3, \delta_{12}^7, \delta_{12}^{10}, \delta_{12}^6, \delta_{12}^3, \delta_{12}^7, \delta_{12}^{10}, \delta_{12}^6$, and δ_{12}^3.

With the obtained states and the given associated event string, we can specify a sample path shown in Figure 1(a). It is interesting to examine the state trajectory more closely. M arrives at states x_3, x_7, x_{10}, x_6 at time t_1, t_2, t_3, t_4, respectively. At the next four time instants t_5, t_6, t_7, t_8, the same set of states x_3, x_7, x_{10}, x_6 are revisited. This process repeats at the coming four time instants $t_9, t_{10}, t_{11}, t_{12}$, and so forth. Such dynamics are very similar to the oscillation phenomenon occurring in nonlinear dynamic systems. Using Eq. (2) to describe the dynamics, this kind of evolution is $x(t+3) = Fx(t)i(t+1)i(t+2)i(t+3) = x(t)$.

This oscillation-like behavior will lead to serious results; that is, an FSM may fall into "traps" or be locked by itself. If no event in the event set can make the FSM transfer to a state other than this set of states, then the FSM enters a trap composed of states x_3, x_6, x_7, x_{10} or is locked by states x_3, x_6, x_7, x_{10}. If the states x_3, x_6, x_7, x_{10} are reachable from each other, then the FSM runs in this state set forever. More seriously, if the state set is reduced to a set comprising only one state, then the FSM remains in a dead state because it will stay in this state forever and will not transfer to any other state.

In the following section, such situations of trap or self-lock are collectively called logical blocking. A loop structure and lock structure are proposed to formulate the logical blocking.

4 Detection of logical blocking

Based on the difference equation-like model of DFSMs, this section is devoted to modeling and determining the logical blocking of DFSMs by following the ideas and methods of control theory. The concepts of loop and lock structures of FSMs are proposed, and an algebraic criterion is developed to determine whether a DFSM has a loop structure or a lock structure.

4.1 Loop and lock structures of DFSMs

Definition 3. A set of states $x_{j_1}, x_{j_2}, \cdots, x_{j_s} \in X$ and the associated events $e_{j_1}, e_{j_2}, \cdots, e_{j_s} \in E$ are said to be a loop structure of length s if the following equation holds. $Fx_{j_k}e_{j_k} := x_{j_{(k+1)}}, k = 1, 2, \cdots, s-1; Fx_{j_s}e_{j_s} := x_{j_1}$.

Definition 4. A set of states $X' = \{x_{j_1}, \cdots, x_{j_s}\} \subset X$ and the associated event set $E' = \{e_{j_1}, \cdots, e_{j_s}\} \subset E$ are said to be a lock structure if the following conditions hold simultaneously. i) For two different states x_{j_p} and x_{j_q} in the set X', there is an event e' in the event set E' such that $Fx_{j_p}e' = x_{j_q}, (p \neq q)$. ii) For any state $x' \in X'$, there exists no event e' in the event set E' such that $Fx'e' = x'', (x'' \notin X')$. If $|X'| = |E'| = 1$, the lock is said to be a dead-lock; otherwise, a live-lock.

As can be seen from Definitions 3 and 4, or from Figure 1(a), the loop of an FSM has a similar meaning to the concept of the oscillation of dynamic systems in control theory, while the lock of an FSM corresponds to the stable behavior of dynamic systems. However, the physical meanings of the loop and lock for DFSMs are quite different from oscillation and stability for dynamic systems. The fundamental reason for this difference is that the two types of systems are governed by two distinct laws; specifically, linear and nonlinear systems abide by the natural laws that can be formulated by differential or difference equations, while FSM evolution is governed by logical rules that are given by human experiences; these rules cannot be modeled by differential or difference equations. We compare the dynamics represented by loop and lock to the oscillation and stability of nonlinear or linear systems because they have very similar state trajectories when we examine the system dynamics by the curve of the dynamic functions. This similarity further enlightens us to use Eqs. (1) and (2) to study the evolution of the states included in a loop structure.

4.2 Detection of loops

We first introduce several operations on matrices and the associated conclusions, which serve as mathematical tools for the development of logical blocking detection.

Definition 5 (Column equipartition of matrices)**.** The r-column equipartition of a $p \times qr$ matrix M is a partition dividing M into r equal-sized blocks $M = [\mathrm{Blk}_1(M), \cdots, \mathrm{Blk}_i(M), \cdots, \mathrm{Blk}_r(M)]$, where $\mathrm{Blk}_i(M) = [\mathrm{Col}_{(i-1)q+1}(M), \mathrm{Col}_{(i-1)q+2}(M), \cdots, \mathrm{Col}_{iq}(M)]$.

Proposition 4 (Extraction of a block from a matrix)**.** Let M be a matrix of size $p \times qr$; then, $\mathrm{M}\delta_r^i = \mathrm{Blk}_i(M)$, where $\mathrm{Blk}_i(M)$ is the i-th block of the r-column equipartition of M.

Definition 6. In the n-column equipartition of $A_{n \times nm}$, the block containing only one nonzero element 1 is called a loop block of A.

Proposition 5 (Extraction of components from the STP of vectors)**.** For vectors $v_1, v_2, \cdots, v_n \in \Delta_m$, assume that their STP is $\ltimes_{i=1}^n v_i = \delta_{m^n}^s$; then, $v_i = D_{[m^{n-1},m]} \ltimes W_{[m^i,m^{n-i}]} \ltimes \delta_{m^n}^s, (i = 1, 2, \cdots, s)$, where $D_{[m^{n-1},m]}$ is defined as $D_{[m,n]} = \mathbf{1}_m \otimes I_n$.

Proof. Left multiplying both sides of $\ltimes_{i=1}^n v_i = \delta_{m^n}^s$ by $W_{[m^i,m^{n-i}]}$ yields

$$W_{[m^i,m^{n-i}]} v_1 \cdots v_n = W_{[m^i,m^{n-i}]} \delta_{m^n}^s.$$

By Proposition 1, we see that $v_{i+1} \cdots v_n v_1 \cdots v_{i-1} v_i = W_{[m^i,m^{n-i}]} \delta_{m^n}^s$. Left multiplying both sides of the equation by $D_{[m^{n-1},m]}$ results in $D_{[m^{n-1},m]} v_{i+1} \cdots v_n v_1 \cdots v_{i-1} v_i = D_{[m^{n-1},m]} W_{[m^i,m^{n-i}]} \delta_{m^n}^s$, According to Proposition 4, we then obtain $v_i = D_{[m^{n-1},m]} W_{[m^i,m^{n-i}]} \delta_{m^n}^s$. The proof is completed.

Theorem 2 (Existence theorem of loops)**.** A DFSM M contains a loop of length l if and only if G^l contains a loop block in its n-column equipartition, where $G^l = F^l W_{[m^l,n]}$, F is the structure matrix of M, m and n are the numbers of events and states of M,respectively, and $W_{[m^l,n]}$ is a swap matrix given in Proposition 2.

Proof. (Necessity) Assume that the DFSM M has a loop of length l, without loss of generality, say, the loop comprising states $x_{i_0}, x_{i_1}, \cdots, x_{i_{l-1}}$ in order. Denote the associated events causing these transitions

Algorithm 1

For a given DFSM $M = (X, E, f, x_0, x_m)$,$X = \{x_1, x_2, \cdots, x_n\}$,$E = \{e_1, e_2, \cdots, e_m\}$. One can obtain all the loop structures of length l.

Step 1: Obtain the structure matrix F of M from Proposition 3.

Step 2: Obtain the loop matrix G^l of M from Theorem 2 and rewrite G^l as its n-column equipartition: $G^l_d = [\mathrm{Blk}_1(G^l), \cdots, \mathrm{Blk}_i(G^l), \cdots, \mathrm{Blk}_n(G^l)]$.

Step 3: Check whether G^l_d contains a loop block; if it does, build the following set $K = \{k|\text{the } k-\text{th block of the } n-\text{column equipartition of } G^l \text{ is a loop block}\}$. Otherwise, terminate the algorithm, and M has no loop.

Step 4: Compute $\delta^{i_1}_m \delta^{i_2}_m \cdots \delta^{i_l}_m$ for each $k \in K$ according to Proposition 5.

Step 5: Compute $\delta^{k_1}_n := F\delta^k_n \delta^{i_1}_m, \ldots, \delta^{k_l}_n := F\delta^{k_l-1}_n \delta^{i_l}_m$. A loop $x_k \xrightarrow{e_{i_1}} x_{k_1} \xrightarrow{e_{i_2}} x_{k_2} \xrightarrow{e_{i_3}} \cdots \xrightarrow{e_{i_l}} x_{k_l}$ is then found by the obtained states and associated events, where $x_{k_l} = x_k$.

Step 6: Repeat Steps 4 and 5 until all $k \in K$ are executed; then all the expected loops are obtained.

as $i(t+1), i(t+2), \cdots, i(t+l)$, respectively; that is, $i(t+1)$ moves M from state x_{i_0} to x_{i_1}, $i(t+2)$ moves M from state x_{i_1} to x_{i_2}, and so on; finally, $i(t+l)$ returns M to state x_{i_0}. According to the definition of loop structure, if we denote x_{i_0} by $x(t)$ we then have

$$x(t+1) := x_{i_1} = Fx_{i_0} i(t+1), x(t+2) := x_{i_2} = Fx_{i_1} i(t+2), \ldots, x(t+l) := x_{i_0} = Fx_{i_{l-1}} i(t+l) = x(t). \quad (3)$$

It follows from the proof of Theorem 1 and Proposition 2 that

$$x(t+l) = F^l x_{i_0} i(t+1) \cdots i(t+l) = F^l x(t) \cdots i(t+l) = F^l W_{[m^l,n]} i(t+1) \cdots i(t+l) x(t) = G^l I(t+l) x(t), \quad (4)$$

where $I(t+l) = i(t+1)i(t+2)\cdots i(t+l)$.

We then have $G^l I(t+l)x(t) = x(t)$. Let the vector forms of $I(t+l)$ and $x(t)$ be $\delta^{k_1}_{m^l}$ and $\delta^{k_2}_n$, respectively. It follows from Proposition 4 that $\mathrm{Blk}_{k_1}(G^l)\delta^{k_2}_n = \delta^{k_2}_n$. According to Proposition 4 again, $\mathrm{Blk}_{k_1}(G^l)$ is clearly a loop block. The necessity is proved.

(Sufficiency) If G^l contains a loop block in its n-column equipartition, say, the k_1th block, then we assume without loss of generality that the element in the position of k_2th column and k_3th row is 1.

It follows from Proposition 4 that there exists a vector $\delta^{k_1}_{m^l}$ such that the k_1th block of $G^l\delta^{k_1}_{m^l}$ is 1, and there exists a vector $\delta^{k_2}_n$ such that $G^l\delta^{k_1}_{m^l}\delta^{k_2}_n = \delta^{k_3}_n$. Using Proposition 5, $\delta^{k_1}_{m^l}$ can be expressed as an STP of a string of vectors: $\delta^{k_1}_{m^l} = \delta^{i_1}_m \delta^{i_2}_m \cdots \delta^{i_l}_m$. Therefore, $G^l\delta^{i_1}_m\delta^{i_2}_m\cdots\delta^{i_l}_m\delta^{k_2}_n = \delta^{k_3}_n$.

Substituting $G^l = F^l W_{[m^l,n]}$ into the equation and combining Propositions 1 and 2, we have

$$\begin{aligned} F^l W_{[m^l,n]}\delta^{i_1}_m\delta^{i_2}_m\cdots\delta^{i_l}_m\delta^{k_2}_n &= F^l\delta^{k_2}_n\delta^{i_1}_m\delta^{i_2}_m\cdots\delta^{i_l}_m = F^{l-1}(F\delta^{k_2}_n\delta^{i_1}_m)\delta^{i_2}_m\cdots\delta^{i_l}_m = \cdots \\ &= F\cdots F(F(F\delta^{k_2}_n\delta^{i_1}_m)\delta^{i_2}_m)\cdots)\delta^{i_{l-1}}_m\delta^{i_l}_m = \delta^{k_3}_n. \end{aligned} \quad (5)$$

Note that the vectors $\delta^{i_1}_m, \delta^{i_2}_m, \cdots, \delta^{i_l}_m$ can be the vector forms of l events, say, $e_{i_1}, e_{i_2}, \cdots, e_{i_l}$, and that $\delta^{k_2}_n$ and $\delta^{k_3}_n$ can be the vector forms of two states, say x_{i_0} and $x_{i_{l-1}}$. By identifying $F\delta^{k_2}_n\delta^{i_1}_m$ as x_{i_1}, $F(F\delta^{k_2}_n\delta^{i_1}_m)\delta^{i_2}_m$ as x_{i_2}, $F\cdots F(F(F\delta^{k_2}_n\delta^{i_1}_m)\delta^{i_2}_m)\cdots)\delta^{i_l}_m$ as x_{i_l}, it is clear that Eq. (5) defines a path beginning with state x_{i_0} and ending with state x_{i_l} and that the associated input events are $e_{i_1}, e_{i_2}, \cdots, e_{i_l}$. In contrast, Eq. (5) suggests that $x_{i_1} = x_{i_0}$, which means that the path is a loop of length l. The sufficiency is then proved.

From the proof of Theorem 2, the following conclusion is obvious.

Corollary 1 (Sufficient condition of existence of loops)**.** If the element "1" of the loop block described in Theorem 2 is in the k-th column, then the DFSM in Theorem 2 contains a loop of length l beginning with and ending with state x_k. In other words, the state x_k is included in a loop.

Remark 3. Theorem 2 and Corollary 1 show again that the dynamic Eqs. (1) and (2) completely describe the information of state evolution in the future t time step, and that the matrix G^l in Theorem 2 contains the complete information of the loops of a DFSM, as we shall see later in Algorithm 1, which finds all the loops of a DFSM using G^l. This attribute leads to defining the matrix G^l as the loop matrix of DFSMs.

Definition 7 (Loop matrix of DFSMs)**.** The matrix G^l in Theorem 2 is called the loop matrix of the DFSM M. As seen from Definitions 3 and 4, a lock structure is necessarily a loop one, but not vice-versa. In contrast, a loop is a relaxed version of a lock, which implies that locks might be found by finding loops. This possibility leads us to consider the issue of how to find all the loops of a DFSM.

Algorithm 1' Improved version of Algorithm 1

Steps 1 to 4 are identical to those of Algorithm 1.
Step 5: Choose a $k \in K$ and calculate $\delta_m^{i_1}, \delta_m^{i_2}, \delta_m^{i_1}, \cdots, \delta_m^{i_l}$ by Proposition 5.
Step 6: Remain constant.
Step 7: Remove $k, k_1, k_2, \cdots, k_l$ from the set K.
Step 8 is identical to Step 7 of Algorithm 1.

Algorithm 2 Detection of locks

Given a DFSM $M = (X, E, f, x_0, X_m)$, one can judge whether M has a lock by performing the following procedure.
Step 1: Set $S_1 = X_m$.
Step 2: Set $S_2 = \varnothing$.
Step 3: Set $S_2 = \{x | \text{there is an } e \in X - S_1 \text{ such that } f(x, e) \in S_1\}$.
Step 4: Set $S_1 = S_1 \cup S_2$.
Step 5: Check whether S_1 is equal to X. If it is, M is non-blocking and the procedure terminates; otherwise, go to Step 6.
Step 6: Check whether S_2 is empty. If it is, M is blocking and the algorithm stops; otherwise, go to Step 2.

Table 1 Operation of Algorithm 2 applied to the DFSM of Example 1

The 1st round	The 2nd round	The 3rd round	The 4th round
Step 1: $S_1 = \{x_9, x_{12}\}$.	Step 2: $S_2 = \varnothing$.	Step 2: $S_2 = \varnothing$.	Step 2: $S_2 = \varnothing$.
Step 2: $S_2 = \varnothing$.	Step 3: $S_2 = \{x_2\}$.	Step 3: $S_2 = \{x_1\}$.	Step 3: $S_2 = \varnothing$.
Step 3: $S_2 = \{x_5\}$.	Step 4: $S_1 = \{x_2, x_5, x_9, x_{12}\}$.	Step 4: S_1=$\{x_1, x_2, x_5, x_9, x_{12}\}$.	Step 4: S_1=$\{x_1, x_2, x_5, x_9, x_{12}\}$.
Step 4: $S_1 = \{x_5, x_9, x_{12}\}$.	Step 5: $S_1 \neq X$, go to Step 6.	Step 5: $S_1 \neq X$, go to Step 6.	Step 5: $S_1 \neq X$, go to Step 6.
Step 5: $S_1 \neq X$, go to Step 6.	Step 6:$S_2 \neq \varnothing$, go to Step 2.	Step 6: $S_2 \neq \varnothing$, go to Step 2.	Step 6: $S_2 = \varnothing$, M is blocking and algorithm stops.
Step 6: $S_2 \neq \varnothing$, go to Step 2.			

5 Search of loops and locks

Thus far, we have formulated the behavior of DFSMs as a difference equation-like model, which is more convenient for studying some particular issues, such as the logical blocking of DFSMs, and have established a criterion for detecting the loops. Now, we can search a DFSM for loops and locks. Finding them, which is achieved in this section via algorithms, helps further overcome the logical blocking of DFSMs.

Observe that any state in a loop can be the starting point and ending point of the loop; therefore, most of the repeats of Steps 5 and 6 of Algorithm 1 are without avail. To avoid this useless effort, Algorithm 1 is modified as Algorithm 1'.

Corollary 2 (The number of loops contained in a DFSM)**.** The DFSM in Algorithm 1 contains $|K|$ loops of length l, where K is the set constructed in Step 4 of Algorithm 1.

Using Theorem 2 and Algorithm 1 (or Algorithm 1'), we can now address the problem of finding all the locks of a given DFSM. To begin with, we present a property of lock structure, which is used to judge whether a DFSM contains a lock.

Proposition 6 (Property of lock structure)**.** Let D be the set of states constituting a lock; then, there is no state $x_2 \in (X - D)$ such that $f(x_1, e) = x_2$ for any $e \in E$ and any state $x_1 \in D$.

The implementation process of Algorithm 2 is collected in Table 1.

Example 1. Consider the DFSM shown in Figure 1(b), where the state set is $X = \{x_1, x_2, \cdots, x_{12}\}$, the event set is $E = \{0, 1\}$, the initial state is $x_0 = x_1$, and accepting state set is $X_m = \{x_9, x_{12}\}$.

Lemma 1 (Location of locks)**.** Let M be a DFSM with a loop (live lock or dead lock), then the loop exists in the set $B = X - S_1$, where S_1 is the set generated by Algorithm 2.

Proof. (Reduction to absurdity) Let D be the set of states forming the loop. If the argument does not hold, that is, there is a state x_1 in $S_1 \cap D$, then according to Proposition 6, no state transition from state x_1 to state x_2 is not in D; therefore, no state transition to states is in S_1. However, it follows from Step 3 of Algorithm 2 that $x_1 \notin S_1$. This result contradicts the fact that $x_1 \in S_1 \cap D$. Then, the statement is correct.

It is easy to use Algorithm 2 to judge whether a DFSM is blocking; however, the loops, including deadlocks and live-locks, are difficult to find. Now, if a DFSM has loops, we try to find all of them.

Example 2. Consider the DFSM in Example 1, where the set S_1 generated by Algorithm 2 is $\{x_1, x_2, x_5, x_9, x_{12}\}$. The operation of Algorithm 3 is shown in Table 2.

Remark 4. The set $B = X - S_1$ of Step 1 of Algorithm 3 can also be replaced by $B = X$. In this situation, the time complexity of computing will increase considerably because of the need for ap-

Algorithm 3 Detection and search of loops

For M given in Algorithm 2, following the steps below, one can judge whether M contains loops and obtain all of them if it does.
Step 1: Apply Algorithm 2 to and set $B = X - S_1 = \{x_{i_1}, x_{i_2}, \cdots, x_{i_s}\}$.
Step 2: Set $L = \varnothing$.
Step 3: Check whether B is empty. If it is, the algorithm terminates and the set of all the loops is L; otherwise, go to Step 4.
Step 4: Set $l = |B|$.
Step 5: Apply Algorithm 1 to each state in B to check whether the state is contained in a loop. If it is, assume that the loop comprises the following set of states: $N = \{x_{i_{j_1}}, x_{i_{j_2}}, \cdots, x_{i_{j_l}}\}$; set $L = L \cup N$, $B = B - N$, and go to Step 4. Otherwise, set $l = l - 1$ and go to Step 5 if $l \neq 0$; if $l = 0$, set $N = \{x_{i_1}\}$.

Table 2 Running process of Algorithm 3 applied to the DFSM of Example 2

The 1st round of running	The 2nd round of running	The 3rd round of running
Step 1: $B = X - S_1 = \{x_3, x_4, x_6, x_7, x_8, x_{10}, x_{11}\}$.	Step 3: $B \neq \varnothing$, go to Step 4.	
Step 2: $L = \varnothing$.	Step 4: $l = \|B\| = 3$.	
Step 3: $B = \varnothing$, go to Step 4.	Step 5: Apply Algorithm 1	Step 3: $B = \varnothing$, algorithm stops,
Step 4: $l = \|B\| = 7$.	to x_4 and obtain that x_4 is in a	and the set of loops of M is $L =$
Step 5: Applying Algorithm 1 to x_3, the result is that x_3 is	loop of length 3: $x_4 \to x_{11} \to$	$\{\{x_3, x_7, x_{10}, x_6\}, \{x_4, x_{11}, x_8\}\}$,
not in a loop of length 7. Set $l = 6$ and repeat Step 5 until	$x_8 \to x_4$. Set $N = \{x_4, x_{11}, x_8\}$,	which is marked by the chain line
$l = 4$, the result is in a loop of length 4: $x_3 \to x_7 \to x_{10} \to$	$L = \{\{L\}, \{N\}\} =$	in Figure 1(c).
$x_6 \to x_3$. Set $N = \{x_3, x_7, x_{10}, x_6\}$,$L = \{\{L\}, \{N\}\} =$	$\{\{x_3, x_7, x_{10}, x_6\}, \{x_4, x_{11}, x_8\}\}$,	
$\{\{x_3, x_7, x_{10}, x_6\}\}$,$B = B - N = \{x_4, x_8, x_{11}\}$, go to Step 3.	$B = B - N = \varnothing$, go to Step 3.	

plying Algorithm 1 to all the states in the entire state set X other than the states in $X - S_1$, where applying Algorithm 1 to the states contained in a lock is repetitive work. However, there is a reward that Algorithm 3 can capture all the loops, including sub-loops (which are included in a loop). Let us examine the DFSM M in Example 2 in more detail. Observe that a shorter loop of length 3, $x_3 \xrightarrow{0} x_7 \xrightarrow{0} x_6 \xrightarrow{1} x_3$ (which is marked by the black dotted line in Figure 1(c)), is contained in the loop $x_3 \xrightarrow{0} x_7 \xrightarrow{1} x_{10} \xrightarrow{1} x_6 \xrightarrow{1} x_3$; a self-loop $x_{11} \xrightarrow{0} x_{11}$ (shown in Figure 1(c), marked by the dotted line) is included in the loop $x_4 \xrightarrow{0} x_{11} \xrightarrow{1} x_8 \xrightarrow{1} x_4$. In addition to its computational complexity, Algorithm 3 is not designed to produce the sub-loops because, in practical applications, one pays more attention to the longer loops because of the inclusion relation between the longer and shorter ones.

6 Analysis of computational complexity

This subsection analyzes the computational complexity of the presented algorithms in terms of time and space in the worst situation. Consider a DFSM $M = (X, E, f, x_0, X_m)$, where $X = \{x_1, x_2, \cdots, x_n\}$ and $E = \{e_1, e_2, \cdots, e_m\}$.

Complexity of Algorithms 1 and 1'. Time complexity: The loop iterations occur in Steps 4, 5, and 6 of Algorithm 1. Step 4 is executed n times in the worst situation, where the n-column equipartition of G^l has n blocks of the same size as a loop block. For the same reason, the set K contains at most n elements. Consequently, Steps 5 and 6 repeat n times in the worst case. Therefore, the loop iterations of Algorithm 1 are $3n$ times in the worst situation, and the time complexity is $O(n)$.

Space complexity: The storage allocation occurs in Steps 1, 2, 4, 5, and 6. Note that although Step 7 involves loop iterations, no more storage is needed. The storage cells required by Steps 1, 2, 4, 5, and 6 are n^2m (used to store the matrix F), n^2m^l (to store the matrix G^l), n (to store the set K), lm (to store the l vectors of $1 \times m$, $\delta_m^{i_1}, \delta_m^{i_2}, \cdots, \delta_m^{i_l}$), and $l \cdot n$ (to store the l vectors of $1 \times n$, $\delta_n^{k_1}, \delta_n^{k_2}, \cdots, \delta_n^{k_l}$), respectively. Therefore, the total memory space allocated to Algorithm 1 is $n^2(m + m^l) + n(l + 1) + m$, and the space complexity is $O\left(n^2(m + m^l)\right)$.

It follows from their differences (i.e., Steps 5 and 7) that Algorithm 1' and Algorithm 1 have identical time and space complexities because the worst time and space complexities are considered. The computational gains of Algorithm 1' lie in the average time complexity.

Complexity of Algorithm 2. Time complexity: Steps 2 to 6 form a circulation that iterates $n - 1$ times in the worst case, where the DFSM is non-blocking. Therefore, the time complexity is $O(n)$.

Space complexity: Step 1 needs at most n^2 storage cells to store the set X_m in the worst case that the set of accepting states is simply the state set of the DFSM. Step 2 consumes no storage. In Step 3, the worst situation occurs when the DFSM has only one accepting state, and all of the remaining states can reach the accepting state; this case requires n^2 storage cells. Consequently, at most $2n^2$ storage cells are needed in Step 4. The remaining steps, Steps 5 and 6, demand no storage to store information. To

summarize, Algorithm 2 needs at most $4n^2$ memory cells, and then the space complexity is $O(n^2)$.

Complexity of Algorithm 3. Time complexity: The time consumption of Algorithm 3 is mainly due to Steps 1 and 5, which use Algorithms 2 and 3 to perform specific operations. Step 5 involves a circulation that executes n times in the worst case, in which the set B contains n states, and none of these states are included in a loop structure. Therefore, according to the time complexities of Algorithms 2 and 3, the time complexity of Algorithm 3 is $O(n^2)$.

Space complexity. By analyzing the space complexity of Algorithm 2, the space complexity of Step 1 is $O(n^2)$. Steps 2, 3, and 4 consume no storage. According to the storage required by Algorithm 1, the space complexity of Step 5 is $O\left(n^2(m+m^l)\right)$. Therefore, the space complexity of Algorithm 3 is $O\left(n^2(m+m^l)\right)$.

Remark 5. Before the STP theory, most methods for studying FSMs were based on traditional methods, such as table lookup, transition graph, and discrete function methods. These methods have the advantages of simplicity, intuitiveness, and clear physical meaning. However, they have a fatal flaw: They cannot describe the dynamic evolution of FSMS in an equation form, and they cannot mathematically investigate the dynamic behaviors. In contrast, the STP method models the discrete, event-driven dynamic behavior of FSMs as bilinear dynamic equations. This approach allows for FSM research within the control theory framework. Therefore, the STP method is advantageous because it can i) provide a quantitative description and prediction of logical blocking behaviors, ii) comprehensively consider logical interactions between multiple factors and variables, and iii) understand the mutual influence mechanisms between the various variables of FSMs. Much like a double-edged sword, the drawback of the STP method is that the proposed algorithms require a large amount of storage space for computer-aided processing when the length of the input sequence being processed is long (longer than 20, according to [22]). Fortunately, this shortcoming may soon be overcome to a great extent with the rapid development of computing speed and storage technology of computers.

7 Further discussions in the context of networks of FSMs

The results of the preceding sections apply to DFSMs, not to nondeterministic FSMs (NFSMs) and thus not to networks of FSMs (networks of FSMs are complex NFSMs). In this section, we further discuss the loop structure and lock structure of networks of FSMs. It is reasonable to start by extending the presented results for DFSMs to the case of NFSMs.

7.1 Results extended to NFSMs

We begin by showing that the dynamic equations of DFSM proposed in [15, 17] also apply to NFSMs after a slight modification. Recall that (i) the vector form of state $x(t)$ in Proposition 3 is a logical vector in which only one element is 1 and the others are 0s, and thus the state $x(t)$ represents a single state. (ii) The key difference between the dynamics of deterministic and nondeterministic FSMs is the result of a state transition, a single state in DFSMs and multiple states in NFSMs. Inspired by these points, a question is naturally posed: "Can the state symbol $x(t)$ in Proposition 3 represent a set of states?" The answer is yes. In fact, in the process of vectorizing FSMs, we can identify a set of states $x_{i_1}, x_{i_2}, \cdots, x_{i_k}$ as the vector $[0, \cdots, 0, 1, 0, \cdots, 0, 1, 0, \cdots, 0, 1, 0, \cdots, 0]$, where the elements at positions $i_1, i_2, \cdots, i_k$ are 1s, and the others are 0s. Note that the vector is not a logical vector, which raises another question: "For nonlogical vectors, is the dynamic equation described in Proposition 3 still correct? " The answer is still yes because the definition of the STP of matrices (Definition 2) involves any matrix, naturally including any type of vector. Therefore, we propose the unified dynamic equation of FSMs, including deterministic and nondeterministic FSMs. Because a DFSM is a special case of nondeterministic FSM, the unified version of the dynamic equation is stated in terms of NFSMs.

Theorem 3 (State evolution equation of NFSMs)**.** The state evolution of an NFSM M driven by an event sequence $u(t) = e_1 e_2 \cdots e_t$ at initial state x_0 is $x(t+1) = F^t x_0 u(t)$, where these objects have the same interpretation as in Proposition 3, with the difference that the vector form of state $x(t)$ is a nonlogical vector described immediately above.

With Theorem 3 as a foundation, the presented results of DFSMs can be easily extended to the case NFSMs, which are summarized as follows.

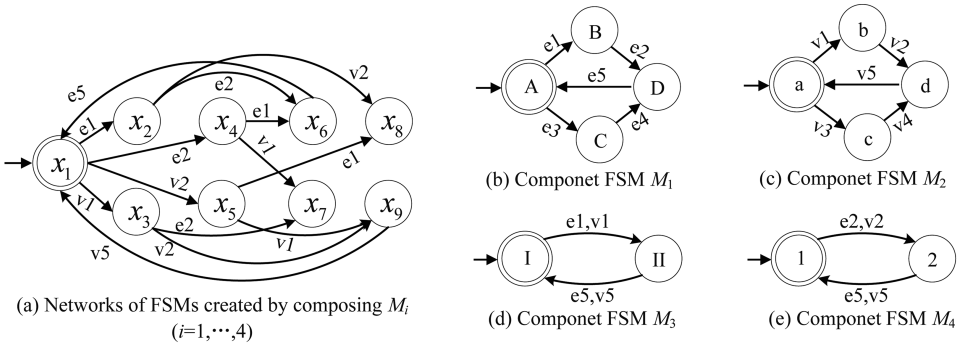

Figure 2 Networks of FSMs and the component FSMs.

Theorem 1 still holds. Notably, the state $x(t)$ is not a logical vector any longer but a nonlogical vector described in Theorem 3. The concepts of loop and lock structures, i.e., Definitions 3 and 4, need no modification, which does not affect the subsequent results that are associated with these two concepts, such as those of the state trajectory of NFSMs. Theorem 2 and Corollary 1 remain correct as long as the loop block of a matrix given in Definition 6 is appropriately extended.

Definition 8 (Extended loop block of a matrix)**.** An extended loop block of $A_{n\times nm}$ is a block of the n-column equipartition of A in which only one column contains 1 (or 1s); the other columns are zero vectors.

Consequently, the remaining results built on Theorem 2 and Corollary 1, such as Algorithms 1, 1', 2, and 3, Lemma 1, Proposition 6 and the complex analysis of these algorithms, still apply to NFSMs.

7.2 Applicability to networks of FSMs

Loop and lock structures are important in the theory of FSMs because they reflect the liveness of a system of FSMs, particularly networks of FSMs. A network of FSMs is a set of FSMs that interact through an operation of composition, called parallel composition, which is defined as follows.

Consider two FSMs $M_1 = (X_1, E_1, f_1, x_{01}, X_{m1})$ and $M_2 = (X_2, E_2, f_2, x_{02}, X_{m2})$. The parallel composition of M_1 and M_2 is the FSM $M_1||M_2 = (X_1 \times X_2, E_1 \cup E_2, f, (x_{01}, x_{02}), X_{m1} \times X_{m2})$, where

$$f\left((x_1,x_2),e\right)=\begin{cases}\left(f_1(x_1,e),f_2(x_2,e)\right), & \text{if both } f_1(x_1,e) \text{ and } f_2(x_2,e) \text{ are defined;}\\ \left(f_1(x_1,e),x_2\right), & \text{if only } f_1(x_1,e) \text{ is defined;}\\ \left(x_1,f_2(x_2,e)\right), & \text{if only } f_2(x_2,e) \text{ is defined;}\\ \text{undefined}, & \text{otherwise.}\end{cases}$$

The parallel composition is associative:$(M_1||M_2)||M_3 = M_1||(M_2||M_3)$. Therefore the parallel composition of a set of n FSMs can be defined by using associativity as

$$W = M_1||M_2||\cdots||M_n = (\cdots((M_1||M_2)||M_3)\,||\cdots M_{n-1})\,||M_n.$$

As seen from the definition, a network of FSMs is large-scale, and it is generally difficult to discover a loop or a lock in a network of FSMs. Finding all the loops and locks of a network of FSMs is a challenging problem. Fortunately, the presented results of FSMs are suitable for this problem because a network of FSMs is, in essence, an NFSM (or a deterministic one in some special cases). For example, the following FSM W shown in Figure 2(a) is a network of four FSMs, M_1, M_2, M_3, and M_4, shown in Figures 2(b)–(e), where the states $x_i (i = 1, 2, \cdots, 9)$ of W are simplified representations of the states of the network for the sake of readability; x_1, for example, stands for the state $(A, a, I, 1)$.

All the loops and locks of the network of FSMs can be found by following the procedure described in Example 2. They are

$$\begin{gathered}\text{Loop 1: } x_1 \xrightarrow{e_1} x_2 \xrightarrow{e_2} x_6 \xrightarrow{e_5} x_1;\ \text{Loop 2: } x_1 \xrightarrow{v_1} x_3 \xrightarrow{v_2} x_9 \xrightarrow{v_5} x_1;\\ \text{Loop 3: } x_1 \xrightarrow{e_2} x_4 \xrightarrow{e_1} x_6 \xrightarrow{e_5} x_1;\ \text{Loop 4: } x_1 \xrightarrow{v_2} x_5 \xrightarrow{v_1} x_9 \xrightarrow{v_5} x_1;\\ \text{Lock 1 (a dead lock): } x_7;\ \text{Lock 2 (a dead lock): } x_8.\end{gathered}$$

Remark 6. The example indicates that the extended results also apply to nondeterministic FSMs. Thus, the results presented in this paper contribute to a deeper mathematical understanding of the logical blocking in FSMs, including both deterministic and nondeterministic FSMs, particularly the logical blockings in large-scale FSMs. However, an open question remains: when multiple FSMs operate simultaneously to accomplish a task, that is, to form a network, is there a mathematical description that characterizes the relation between the logical behavior of the component FSMs and that of the entire network? If so, how to describe it, and what is the description? If such a description can be established, the ideas, methods, and results proposed in this paper could be applied at a deeper level to study logical blocking problems of networks of FSMs.

8 Conclusion

The dynamic behaviors of FSMs have two types of blocking: time blocking and logical blocking. Logical blocking can damage the liveness of FSMs. This paper examines the liveness issue of FSMs from a new perspective. By viewing an FSM as a logical dynamic system, the liveness of FSMs is considered from the aspects of modeling dynamics and detecting and searching the logical blocking of an FSM. The idea of stability and the method for drawing the state trajectory of linear systems are borrowed. The input event strings are treated as the input signals of the logical system, and the "states" of an FSM are simply considered the system states. To model blocking, a loop structure and a lock structure are proposed to represent two types of blocking, live-lock and dead-lock. For the detection problem, a criterion is proposed to judge whether an FSM is blocking. The problem of searching for the logical blocking of an FSM is addressed by developing search algorithms, which can find all the logical blocking of an FSM if it has logical blocking. Furthermore, these results are extended to apply to nondeterministic FSMs and, thus, to networks of FSMs. We believe that the proposed STP approach provides a new angle for considering many problems in the area of FSMs, such as system analysis and model checking.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (Grant Nos. U1804150, 62073124).

References

1 Cassandras C G, Lafortune S. Introduction to Discrete Event Systems. 2nd ed. New York: Springer, 2008
2 Prigioniero L. Regular Languages: To Finite Automata and Beyond Succinct Descriptions and Optimal Simulations. European Association for Theoretical Computer Science, 2020
3 Xu X R, Hong Y G. Observability analysis and observer design for finite automata via matrix approach. IET Control Theor Appl, 2013, 7: 1609–1615
4 Yan Y Y, Cheng D Z, Feng J-E, et al. Survey on applications of algebraic state space theory of logical systems to finite state machines. Sci China Inf Sci, 2023, 66: 111201
5 Yan Y Y, Chen Z Q, Yue J M. Algebraic state space approach to model and control combined automata. Front Comput Sci, 2017, 11: 874–886
6 Li Y L, Feng J-E, Cheng D Z, et al. Observability decomposition of Boolean control networks under several kinds of observability. IEEE Trans Automat Contr, 2023, 68: 1–8
7 Barkalov A, Titarenko L. Background of finite state machines and programmable logic. In: Logic Synthesis for FPGA-Based Finite State Machines, Springer International Publishing, 2016. 1–31
8 Cheng D Z, Qi H S, Zhao Y. An Introduction to Semi-Tensor Product of Matrices and Its Applications. Singapore: World Scientific Publishing Co. Pte. Ltd., 2012
9 Zhu S Y, Cao J D, Lin L, et al. Toward stabilizable large-scale Boolean networks by controlling the minimal set of nodes. IEEE Trans Automat Contr, 2024, 69: 174–188
10 Wu J H, Liu Y, Ruan Q H, et al. Robust stability of switched Boolean networks with function perturbation. Nonlinear Anal-Hybrid Syst, 2022, 46: 101216
11 Wang Y H, Cheng D Z. On coset weighted potential game. J Franklin Inst, 2020, 357: 5523–5540
12 Wu Y H, Le S T, Zhang K Z, et al. Ex-ante agent transformation of Bayesian games. IEEE Trans Automat Contr, 2022, 67: 5793–5808
13 Yan Y Y, Deng H, Yue J M. Model-reference adaptive control of finite state machines with respect to states: a matrix-based approach. IEEE Trans Circ Syst II, 2023, 70: 2171–2175
14 Xu X R, Hong Y G. Matrix expression and reachability analysis of finite automata. J Control Theor Appl, 2012, 10: 210–215
15 Li B W, Lu J Q. Boolean-network-based approach for construction of filter generators. Sci China Inf Sci, 2020, 63: 220–232
16 Li H T, Yang X R. Robust optimal control of logical control networks with function perturbation. Automatica, 2023, 152: 110970
17 Wu Y H, Zhang J Y, Shen T L. A logical network approximation to optimal control on a continuous domain and its application to HEV control. Sci China Inf Sci, 2022, 65: 212203
18 Yan Y Y, Xu P J, Yue J M, et al. Robust control: from continuous-state systems to finite state machines. IEEE Trans Autom Sci Eng, 2024, 21: 2156–2163
19 Cheng D Z, Qi H S. A linear representation of dynamics of Boolean networks. IEEE Trans Automat Contr, 2010, 55: 2251–2258
20 Yue J M, Yan Y Y, Chen Z Q. A simplifying-matrix method to reduce the storage complexity in studying finite state machines in the framework of STP. IEEE Trans Circ Syst II, 2023, 70: 196–200
21 Yan Y Y, Chen Z Q, Liu Z X. Semi-tensor product approach to controllability and stabilizability of finite automata. J Syst Eng Electron, 2015, 26: 134–141
22 Cheng D Z, Qi H S, Li Z Q. Model construction of Boolean network via observed data. IEEE Trans Neural Netw, 2011, 22: 525–536