SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

April 2026, Vol. 69, Iss. 4, 140203:1–140203:17
https://doi.org/10.1007/s11432-025-4788-y

Special Topic: Logical System Control

A data-driven framework for constrained control of Boolean networks

Dingyuan ZHONG[1], Yuanyuan LI[2], Jie ZHONG[3], Ping ZHANG[4] & Jianquan LU[1*]

[1]*School of Mathematics, Southeast University, Nanjing 210096, China*
[2]*Department of Applied Mathematics, Nanjing Forestry University, Nanjing 210018, China*
[3]*College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China*
[4]*Department of Automation Technology and Complex Systems, University of Duisburg-Essen, Essen 45117, Germany*

Received 4 October 2025/Revised 8 December 2025/Accepted 18 December 2025/Published online 10 March 2026

Abstract In this paper, a new data-driven framework is proposed for controller design of Boolean networks, without requiring prior knowledge of the network structure. A key advantage is its capability to handle various state, input, or performance constraints that commonly arise in practical control tasks while ensuring stabilization. Unlike linear systems, generating sufficient data for Boolean networks presents unique challenges. To address this, an implementable method is developed to construct suitable input sequences that guarantee data richness for controller design. The advantage and applicability of our framework are demonstrated through two representative scenarios, namely stabilization under temporal tasks and stabilization with optimal control, for illustration. Finally, the theoretical results are validated through a biological example.

Keywords Boolean networks, data-driven control, stabilization, algebraic state space representation

Citation Zhong D Y, Li Y Y, Zhong J, et al. A data-driven framework for constrained control of Boolean networks. Sci China Inf Sci, 2026, 69(4): 140203, https://doi.org/10.1007/s11432-025-4788-y

1 Introduction

Boolean networks (BNs), initially introduced by Kauffman, are discrete-time nonlinear systems that effectively simulate genetic regulatory networks [1]. Each node in a BN is represented by a binary value, and its update depends on logical interactions with neighboring nodes. Due to their simplicity and interpretability (where binary values often represent “on/off”, “active/inactive”, or “high/low”), broad applications of BNs have been found in areas such as smart-building control [2], game theory [3], and finite state machines [4].

A central objective in studying BNs is to understand their dynamics and develop appropriate control strategies, which is essential for controlling biological and engineering systems [5, 6]. To overcome this challenge, the algebraic state space representation (ASSR) approach was formulated based on the semitensor product of matrices [7]. In this setup, the ASSR approach has enabled significant progress in BNs, including stability and stabilization [8–10], controllability and observability [11–13], and reconstruction and identification [14–16].

Notably, most existing results on BNs assume full knowledge of the logical update functions. However, these functions are not always available or easy to identify. While data-driven methods have been extensively studied for linear systems [17–19], only a few attempts have been made for BNs. Building on the concept of data informativity, Refs. [20, 21] designed state-feedback controllers for stabilization. In [20], an effective algorithm is proposed that yields a feasible stabilizer. However, it generates only one controller, which may become invalid when additional constraints are imposed, leaving no alternative solution. To address this limitation, it is desirable to characterize all feasible stabilizers for constrained control problems. This issue is partially addressed in [21], where all feasible ones are constructed when the collected data are informative for stabilizability. Unfortunately, this work offers little direction on how to acquire such informative data.

Motivated by the above discussion, we develop a new data-driven framework for the control of BNs. It explicitly shows how to obtain sufficient data for analysis, and, based on it, how to derive all feasible controllers. This framework is particularly suitable for constrained and multi-objective control problems in practical systems, such

* Corresponding author (email: jqluma@seu.edu.cn)

as cell differentiation [22] and unmanned aerial vehicles [23], as it enables systematic construction of a controller that best meets given requirements.

While these advantages demonstrate its potential for broad applicability, applying the framework in practice raises several challenges. First, it is crucial to determine whether the available data are sufficient for controller design. Constrained control problems require data that are both rich (sufficient in quantity) and well-distributed (covering the relevant state-input space); otherwise, the resulting controller may violate constraints or fail to achieve desired performance. Based on this, the second challenge involves how to generate such data systematically. While data richness in linear systems can be guaranteed by standard results such as Willems et al.'s fundamental lemma [24], these results do not apply to BNs, whose state transitions follow discrete logical rules rather than algebraic linear relations. Consequently, richness cannot be ensured by linear-algebraic constructions, and a method tailored to BNs is required.

To overcome these difficulties, the main contributions of this paper are shown as follows.

- We develop a new data-driven control framework that constructs all feasible stabilizing controllers directly from collected data, without requiring the logical functions. It also supports an event-triggered implementation to significantly reduce control updates, and serves as a foundation for constrained control problems.
- We establish systematic procedures for generating sufficient data within our framework. Unlike existing methods for linear systems, BNs require explicitly designed input sequences, and our method provides a practical guideline for collecting such data.
- We demonstrate how the framework can address constrained control problems, including state/input limitations, performance requirements, or task specifications. Two representative scenarios, stabilization under temporal tasks and stabilization with optimal control, are presented for illustration.

The rest of the paper is organized as follows. Section 2 introduces the problem setting and fundamental preliminaries. Section 3 presents a data-driven parameterization of BNs. In Section 4, we derive the main theoretical results and propose a feasible method to design stabilizers. Section 5 deals with the constrained control problems. Section 6 illustrates our results with a biological example. Finally, Section 7 concludes the paper.

Notations. Denote by $\mathbb{N}$ and $\mathbb{R}$ the set of all nonnegative and real numbers, respectively. $[a:b] := [a,b]\cap\mathbb{N}$. $\Delta_k := \{\delta_k^i \mid i \in [1:k]\}$, with δ_k^i denoting the ith column of the $k\times k$ identity matrix I_k. Define the set of logical matrices as $\mathscr{L}^{m\times n} := \{M \mid M\in\mathbb{R}^{m\times n}, \mathrm{Col}(M)\subset\Delta_m\}$, where $\mathrm{Col}(M)$ is the set of column vectors of M. For $M=[\delta_m^{i_1},\delta_m^{i_2},\dots,\delta_m^{i_n}]\in\mathscr{L}^{m\times n}$, denote $M=\delta_m[i_1,i_2,\dots,i_n]$ for brevity. Let $A^\dagger$ denote the Moore-Penrose generalized inverse matrix of $A\in\mathbb{R}^{m\times n}$. The binary domain is denoted by $\mathscr{D}:=\{0,1\}$, and $\mathscr{D}^n=\underbrace{\mathscr{D}\times\mathscr{D}\times\cdots\times\mathscr{D}}_{n}$. For a logical matrix $M\in\mathscr{L}^{m\times n}$, its pth power M^p is defined as the p-fold semitensor product $M^p=\underbrace{M\ltimes\cdots\ltimes M}_{p}$.

2 Preliminaries and framework

2.1 BCNs and the algebraic form

In this subsection, we will introduce the BCN model and its algebraic state-space form based on the semitensor product.

Consider a typical BCN with n state nodes and m control inputs, which is formulated as

$$x_i(t+1)=f_i(U(t),X(t)),\quad i\in[1:n], \tag{1}$$

where $X(t)=(x_1(t),\dots,x_n(t))\in\mathscr{D}^n$ and $U(t)=(u_1(t),\dots,u_m(t))\in\mathscr{D}^m$ are the system state and control input at time $t\in\mathbb{N}$, with $x_i\in\mathscr{D},\ i\in[1:n]$ being state of the ith node, and $u_j(t),\ j\in[1:m]$ being the jth controller. $f_i:\mathscr{D}^{m+n}\mapsto\mathscr{D},\ i\in[1:n]$ is the logical function describing the nodal dynamics, which is constant and unknown.

The objective is to develop a state-feedback control strategy to stabilize system (1). In particular, the dynamics of the controller are given by

$$u_j(t)=k_i(X(t)),\ j\in[1:m], \tag{2}$$

where $k_j:\mathscr{D}^n\mapsto\mathscr{D}^m,\ j\in[1:m]$ is the logical function.

To transform the above system description into an algebraic state-space form, we now introduce the concept of semitensor product of matrices.

Definition 1 ([7]). The semitensor product of matrices $M\in\mathbb{R}^{m_1\times m_2}$ and $N\in\mathbb{R}^{n_1\times n_2}$ is defined as

$$M\ltimes N=(M\otimes I_{l/m_2})(N\otimes I_{l/n_1}),$$

where $l = \mathrm{lcm}(m_2, n_1)$ denotes the least common multiple of m_2 and n_1, and "$\otimes$" refers to the Kronecker product of matrices.

To apply the semitensor product in the analysis of BNs, define a bijective mapping that transforms Boolean vectors into their equivalent logical vector form. Specifically, for $X = (x_1, \ldots, x_n) \in \mathscr{D}^n$, define a bijection $\varphi_n : \mathscr{D}^n \mapsto \Delta_{2^n}$ such that

$$x := \varphi_n(X), \tag{3}$$

where $x \in \Delta_{2^n}$ and

$$\varphi_n(X) = \begin{bmatrix} x_1 \\ 1 - x_1 \end{bmatrix} \ltimes \begin{bmatrix} x_2 \\ 1 - x_2 \end{bmatrix} \ltimes \cdots \ltimes \begin{bmatrix} x_n \\ 1 - x_n \end{bmatrix}.$$

This transformation allows logical functions to be converted into their equivalent algebraic forms, as stated in Lemma 1.

Lemma 1 ([7]). For a logical function $f(X) : \mathscr{D}^n \mapsto \mathscr{D}$, there exists a unique matrix $M_f \in \mathscr{L}^{2\times 2^n}$, referred to as the structure matrix of f, such that

$$\varphi_1(f(X)) = M_f \varphi_n(X).$$

Definition 2 ([25]). The Khatri-Rao product of matrices $M \in \mathbb{R}^{m\times p}$ and $N \in \mathbb{R}^{n\times p}$ is defined as

$$M * N = [\mathrm{Col}_1(M) \ltimes \mathrm{Col}_1(N), \ldots, \mathrm{Col}_p(M) \ltimes \mathrm{Col}_p(N)].$$

Based on Definitions 1 and 2 and Lemma 1, BCN (1) and controller (2) can be equivalently rewritten in an algebraic form

$$x(t+1) = Fu(t)x(t), \tag{4}$$

and

$$u(t) = Kx(t), \tag{5}$$

where $x(t) = \varphi_n(X(t)) \in \Delta_{2^n}$ and $u(t) = \varphi_m(U(t)) \in \Delta_{2^m}$; F and K are calculated as $F = F_1 * \cdots * F_n$ and $K = K_1 * \cdots * K_m$ with F_i and K_j being the structure matrices of f_i and k_j, respectively.

2.2 Data collection

Since $f_i, i \in [1 : n]$ are unknown, classical control methods for BCNs (e.g., [26,27]) are not directly applicable. The primary challenge, therefore, is to design a controller based solely on the available data. Specifically, assume that an offline dataset has been collected from an experiment in the following form:

$$\mathbb{D} = \{U(t), X(t), X(t+1) \mid t \in [0 : T-1]\}, \tag{6}$$

where $X(t) \in \mathscr{D}^n$ and $U(t) \in \mathscr{D}^m$ are the system state and control input collected at time $t \in \mathbb{N}$, respectively.

To facilitate subsequent analysis, we convert (6) into the logical form as in (3) and reorganize the collected data into matrix form

$$U_{[0:T-1]} = [u(0), u(1), \ldots, u(T-1)] \in \mathscr{L}^{2^m\times T}, \tag{7a}$$

$$X_{[0:T-1]} = [x(0), x(1), \ldots, x(T-1)] \in \mathscr{L}^{2^n\times T}, \tag{7b}$$

$$X_{[1:T]} = [x(1), x(2), \ldots, x(T)] \in \mathscr{L}^{2^n\times T}, \tag{7c}$$

where $x(t) = \varphi_n(X(t)) \in \Delta_{2^n}$ and $u(t) = \varphi_m(U(t)) \in \Delta_{2^m}$.

The objective is to determine, from the dataset $\mathbb{D}$, a state-feedback gain matrix K in controller (5) such that the resulting closed-loop BCN can be stabilized to a desired state.

3 Data-based dynamics of BCNs

In this section, we focus on system (4) with dataset $\mathbb{D}$ from (6) (or equivalently, (7)). Based on this, we propose a data-driven parameterization for BCNs.

3.1 Assumption on $\mathbb{D}$

To proceed, we impose a condition that ensures the richness of the data.

Assumption 1. The matrix $U_{[0:T-1]} * X_{[0:T-1]}$ has full row rank, that is,

$$\operatorname{rank}\left(U_{[0:T-1]} * X_{[0:T-1]}\right) = 2^{m+n}. \tag{8}$$

Remark 1. Assumption 1 guarantees that the dataset contains all the information required to characterize all feasible state-feedback controllers, which is essential for constrained control problems. To satisfy this, Subsection 4.3 provides a systematic guideline for data collection.

Remark 2. Before checking the rank condition in Assumption 1, the collected dataset can be pre-processed by removing duplicated columns while keeping one copy. This operation does not reduce the matrix rank but allows for a more efficient verification.

For notation simplicity, in the remainder of the paper, we abbreviate $U_{[0:T-1]}$, $X_{[0:T-1]}$ and $X_{[1:T]}$ as U_0, X_0 and X_1, respectively.

3.2 Data-based representation of BCNs

In this subsection, the data-based representation of a BCN is presented. We begin with Lemma 2.

Lemma 2. Given that Assumption 1 is satisfied, $X_1(U_0 * X_0)^\dagger$ is a logical matrix.

Proof. Let $A = U_0 * X_0 \in \mathscr{L}^{2^{m+n}\times T}$. When A has full row rank, its generalized inverse is given by $A^\dagger = A^\top(AA^\top)^{-1}$. Then, its (i,j)th entry can be calculated as

$$\begin{aligned}
[A^\dagger]_{i,j} &= \sum_{l=1}^{2^{m+n}} (A^\top)_{i,l} \cdot [(AA^\top)^{-1}]_{l,j} \\
&= [A]_{j,i} \cdot [(AA^\top)^{-1}]_{j,j} \\
&= \frac{1}{\sum_{k=1}^{T} [A]_{i,k}} (A^\top)_{i,j},
\end{aligned}$$

where the last two equalities follow from the fact that A is logical and $AA^\top$ is a diagonal matrix, where

$$[AA^\top]_{i,j} = \sum_{k=1}^{T} A_{i,k} \cdot A_{j,k} = \begin{cases} \sum_{k=1}^{T} A_{i,k}, & i = j, \\ 0, & i \neq j. \end{cases}$$

We first suppose that Assumption 1 holds with $T = 2^{m+n}$. In this case, $U_0 * X_0$ spans the entire state space $\Delta_{2^{m+n}}$, and thus each row of A contains a single 1, that is $\sum_{k=1}^{T} A_{i,k} = 1, \forall i \in [1 : 2^{m+n}]$. Therefore,

$$A^\dagger = A^\top \in \mathscr{L}^{T\times 2^{m+n}}.$$

For any $X_1 \in \mathscr{L}^{2^n\times T}$, denote $X_1 = \delta_{2^n}[i_1, i_2, \ldots, i_T]$, and let $A^\dagger = \delta_T[j_1, j_2, \ldots, j_{2^{m+n}}]$. A simple calculation yields

$$X_1 A^\dagger = \delta_{2^n}[i_{j_1}, i_{j_2}, \ldots, i_{j_{2^{m+n}}}],$$

which implies that $X_1 A^\dagger$ is a logical matrix.

Next, consider the case where Assumption 1 is first satisfied when $T > 2^{m+n}$. Then, there must be at least one index $j \in [1 : 2^{m+n}]$ such that $\sum_{k=1}^{T} A_{j,k} > 1$. Without loss of generality, assume that the columns of X_0 and U_0 indexed by $\mathscr{P} = \{p_1, p_2, \ldots, p_k\}$ are identical. Denote $\operatorname{Col}_i(X_0) = \delta_{2^n}^p$ and $\operatorname{Col}_i(U_0) = \delta_{2^m}^q$. Then the corresponding columns of A are equal, i.e., $\operatorname{Col}_i(A) = \delta_{2^{m+n}}^{(q-1)2^n+p}, i \in \mathscr{P}$. As a result, the $(i, (q-1)2^n + p)$th entry of $A^\dagger$ is given by

$$[A^\dagger]_{i,(q-1)2^n+p} = \begin{cases} \frac{1}{k}, & i \in \mathscr{P}, \\ 0, & \text{otherwise}. \end{cases}$$

To demonstrate that $X_1 A^\dagger$ is logical, it suffices to prove $\operatorname{Col}_{(q-1)2^n+p}(X_1 A^\dagger) \in \Delta_{2^n}$. In fact,

$$[X_1 A^\dagger]_{l,(q-1)2^n+p} = \sum_{i=1}^{T} [X_1]_{l,i} \cdot [A^\dagger]_{i,(q-1)2^n+p}$$

$$= \sum_{i\in\mathscr{P}} \frac{1}{k}[X_1]_{l,i}$$
$$= \begin{cases} 1, \, l = i_{p_1+1}, \\ 0, \text{ otherwise,} \end{cases}$$

where the last equality holds since all the columns of X_1 indexed by $\mathscr{P}$ are equal as well, due to identical $x(t)$ and $u(t)$ at those indices. Therefore, we conclude that $X_1(U_0 * X_0)^\dagger$ is logical.

This leads to the following theorem, providing the unique data-based representation of BCN (4).

Theorem 1. Let Assumption 1 hold. Then BCN (4) has the following equivalent representation:

$$x(t+1) = X_1(U_0 * X_0)^\dagger u(t)x(t). \tag{9}$$

Proof. Since $A := U_0 * X_0$ has full row rank by Assumption 1, there must exist a vector $g \in \mathbb{R}^T$ such that $Ag = u * x$. One possible g is $g = A^\dagger(u*x) + \Pi_S^\perp w, w \in \mathbb{R}^T$, and $\Pi_S^\perp = I_T - A^\dagger A$ denotes the orthogonal projector onto the null space of A. Then we have

$$\begin{aligned} x(t+1) &= F(u(t) * x(t)) \\ &= F(U_0 * X_0)g(t) \\ &= X_1 g(t), \end{aligned}$$

where the last equality holds since $X_1 = F(U_0 * X_0)$.

Following that,

$$\begin{aligned} x(t+1) &= X_1((U_0 * X_0)^\dagger(u(t) * x(t)) + \Pi_S^\perp w) \\ &= X_1(U_0 * X_0)^\dagger u(t)x(t), \end{aligned}$$

where the last equality holds since $X_1\Pi_S^\perp = FA(I - A^\dagger A) = 0$.

4 Stabilizing BCNs from data

In this section, we explore the stabilizability of BCNs and design state-feedback controllers directly from the available data. Furthermore, we extend the results to the event-triggered scenario to reduce the control cost.

4.1 State-feedback control from data

To begin, we explore the representation of BCN (4) under a state-feedback controller (5).

Theorem 2. Let Assumption 1 hold. Then, the closed-loop BCN (4) under the state-feedback control (5) has the following equivalent representation:

$$x(t+1) = X_1 G_K x(t), \tag{10}$$

where $G_K \in \mathbb{R}^{T\times 2^n}$ satisfies

$$(U_0 * X_0)G_K = K\Phi_n, \tag{11}$$

and $\Phi_n = [\delta_{2^n}^1 \ltimes \delta_{2^n}^1, \delta_{2^n}^2 \ltimes \delta_{2^n}^2, \ldots, \delta_{2^n}^{2^n} \ltimes \delta_{2^n}^{2^n}]$.

Proof. Applying controller (5), BCN (4) becomes a closed-loop system:

$$x(t+1) = FKx(t)x(t) = FK\Phi_n x(t).$$

Since $U_0 * X_0$ has full row rank by Assumption 1, there exists a matrix $G_K \in \mathbb{R}^{T\times 2^n}$, such that

$$(U_0 * X_0)G_K = K\Phi_n.$$

Consequently,

$$x(t+1) = F(U_0 * X_0)G_K x(t) = X_1 G_K x(t).$$

It is worth noting that the matrix $X_1 G_K$ must be a logical matrix, since $X_1 G_K = X_1(U_0 * X_0)^\dagger K\Phi_n$ must be logical by Lemma 2.

Let $x(t; x_0, \mathbf{u})$ denote the system trajectory starting at x_0 under the control sequence $\mathbf{u} = \{u(t) \mid t \in \mathbb{N}\}$. We begin by recalling the definition of stabilizability.

Definition 3. Given a target state x_e, BCN (4) is said to be finite-time x_e-stabilizable if for any $x_0 \in \Delta_{2^n}$, there exist an integer $T > 0$ and a control sequence $\mathbf{u} = \{u(t) \mid t \in \mathbb{N}\}$ such that

$$x(t; x_0, \mathbf{u}) = x_e, \forall t \geqslant T.$$

Reviewing the Lyapunov-based stability analysis of BCNs, we obtain the following result.

Lemma 3 (cf. [26]). BCN (4) is stabilized at x_e under a given control sequence $\mathbf{u}$, if and only if there exists a Lyapunov function $V(x(t))$ of the form

$$V(x(t)) = \sum_{i=1}^{2^n} a_i x_i(t), \tag{12}$$

where $x(t) = [x_1(t), x_2(t), \ldots, x_{2^n}(t)]^\top$ and $a_i \in \mathbb{R}, i \in [1 : 2^n]$, and the following conditions are satisfied:

(1) $V(x(t)) > 0$ for $x(t) \neq x_e$, and $V(x(t)) = 0$ for $x(t) = x_e$;

(2) $\Delta V(x(t)) < 0$ for $x(t) \neq x_e$, and $\Delta V(x(t)) = 0$ for $x(t) = x_e$, where $\Delta V(x(t)) = V(x(t + 1)) - V(x(t))$.

Proof. As shown in [26], the stability of a BCN under a given control sequence is equivalent to the existence of a Lyapunov function

$$V(x) = J_1 M_v x,$$

where $J_1 M_v \in \mathbb{R}^{2^n}$ is a row vector and $x \in \Delta_{2^n}$. Letting $J_1 M_V = [a_1, a_2, \ldots, a_{2^n}]$ yields the expression in (12).

Then, we present the following theorem for stabilization.

Theorem 3. Suppose that Assumption 1 holds. Consider the following set of inequalities:

$$\begin{cases} q_i > 0, \text{ for } i \neq r, \\ q_r = 0, \\ \left[(X_1 G_K)^\top q - q\right]_i < 0, \text{ for } i \neq r, \\ \left[(X_1 G_K)^\top q - q\right]_r = 0. \end{cases} \tag{13}$$

Then the state-feedback gain

$$K = D_{[2^m, 2^n]}(U_0 * X_0) G_K \tag{14}$$

stabilizes BCN (4) at $x_e = \delta_{2^n}^r$ in finite time, if the following conditions are satisfied:

(1) there exist a vector $q \in \mathbb{R}^{2^n}$ and a matrix $G_K \in \mathbb{R}^{T \times 2^n}$ such that (13) is feasible;

(2) $X_1 G_K$ is a logical matrix, and satisfies

$$X_1 G_K = X_1 (U_0 * X_0)^\dagger D_{[2^m, 2^n]}(U_0 * X_0) G_K \Phi_n, \tag{15}$$

where $D_{[2^m, 2^n]} = I_{2^m} \otimes \mathbf{1}_{2^n}^\top$.

Conversely, if BCN (4) is finite-time stabilizable at $x_e = \delta_{2^n}^r$ by a state-feedback gain K, then it can be written as in (14). Moreover, there exist a vector $q \in \mathbb{R}^{2^n}$ and a matrix G_K (satisfying (11)) such that Eq. (13) is feasible.

Proof. [Sufficiency] The proof proceeds in two steps. First, we verify that the closed-loop system is well-defined. Then we show that it achieves stabilization.

From Theorem 2, the closed-loop BCN (4) under state-feedback control can be written as

$$x(t + 1) = X_1 G_K x(t),$$

where $X_1 G_K$ is a logical matrix. Alternatively, applying (14) to BCN (4) and using Theorem 1, one can obtain

$$\begin{aligned} x(t + 1) =& X_1 (U_0 * X_0)^\dagger u(t) x(t) \\ =& X_1 (U_0 * X_0)^\dagger D_{[2^m, 2^n]}(U_0 * X_0) G_K \Phi_n x(t). \end{aligned}$$

To ensure the system is well-defined, these two representations must be consistent, and this requirement is precisely captured by (15).

We now prove finite-time stabilization under condition (1). Suppose there exists $q \in \mathbb{R}^{2^n}$ satisfying (13). Define a Lyapunov function as

$$V(x(t)) = x^\top(t) q. \tag{16}$$

To verify that (16) is well-defined and nonnegative, let $x(t) = \delta_{2^n}^i$. Then,

$$V(x(t)) = (\delta_{2^n}^i)^\top q = q_i \geqslant 0,$$

with equality if and only if $i = r$. From (10), we further obtain

$$\begin{aligned}\Delta V(x(t)) &= x^\top(t)(X_1 G_K)^\top q - x^\top(t) q \\ &= x^\top(t)[(X_1 G_K)^\top q - q] \leqslant 0,\end{aligned} \tag{17}$$

where equality holds if and only if $x(t) = x_e$.

To establish stabilizability, we begin by showing that x_e is a fixed point. Suppose, for contradiction, that $\mathrm{Col}_r(X_1 G_K) = \delta_{2^n}^i$ for some $i \neq r$. Then, for $x(t) = x_e$,

$$\begin{aligned}\Delta V(x(t)) &= [X_1 G_K \delta_{2^n}^r]^\top q - (\delta_{2^n}^r)^\top q \\ &= (\delta_{2^n}^i)^\top q - (\delta_{2^n}^r)^\top q \\ &= q_i > 0,\end{aligned}$$

which contradicts (17). Therefore, x_e must be a fixed point.

Consider a trajectory starting from an arbitrary initial state $x(0) = \delta_{2^n}^{i_0}$ with $i_0 \neq r$:

$$\delta_{2^n}^{i_0} \longrightarrow \delta_{2^n}^{i_1} \longrightarrow \delta_{2^n}^{i_2} \longrightarrow \cdots \longrightarrow \delta_{2^n}^{i_k} \longrightarrow \cdots.$$

The Lyapunov function along this trajectory satisfies

$$q_{i_0} > q_{i_1} > q_{i_2} > \cdots > q_{i_k} > \cdots.$$

Since the state space Δ_{2^n} is finite, the sequence must reach $q_{i_{k_0}} = 0$, i.e., $x(i_{k_0}) = x_e$, in at most $2^n - 1$ steps. As $x(0)$ is arbitrary, the system is finite-time x_e-stabilizable.

[Necessity] By Lemma 3, when BCN (4) is finite-time x_e-stabilizable under the controller $u(t) = Kx(t)$, there exists a Lyapunov function of the form

$$V(x(t)) = \sum_{i=1}^{2^n} q_i x_i(t) = x^\top(t) q,$$

where $q = [q_1, q_2, \ldots, q_{2^n}]^\top$ and it satisfies

$$\begin{cases} V(x_e) = (\delta_{2^n}^r)^\top q = 0, \\ V(x) = (\delta_{2^n}^i)^\top q > 0,\ i \neq r, \\ \Delta(x(t)) = x^\top((X_1 G_K)^\top - I) q < 0,\ x \neq x_e, \\ (\delta_{2^n}^r)^\top ((X_1 G_K)^\top - I) q = 0. \end{cases}$$

Once G_K is determined by (11), the controller can be expressed in a data-based form as

$$u(t) = D_{[2^m, 2^n]} u(t) x(t) = D_{[2^m, 2^n]} (U_0 * X_0) G_K x(t),$$

which completes the proof.

Remark 3. Theorem 3 provides a data-driven framework to construct all feasible stabilizing controllers, which is a necessary first step for designing controllers that optimize performance or satisfy additional constraints. Condition (1) guarantees stabilizability, while condition (2) ensures that the closed-loop system is uniquely defined. To be more specific, the state transition of BCN (4) can be represented either by the state-feedback gain (14) or the data-based form (10). Their consistency ensures a well-defined next state for any $x(t)$.

We now discuss practical methods for verifying Theorem 3.

For any given K, the stabilizability of BCN (4) at x_e can be determined by checking (13), which is transformed into a linear programming feasibility problem, given as follows.

Proposition 1. A given state-feedback gain K can stabilize BCN (4) at x_e in finite time if and only if Algorithm 1 returns a feasible solution $q \in \mathbb{R}^{2^n}$.

Algorithm 1 Determine whether a given K can stabilize a BCN.

```
Input: $X_1G_K$ and $r$.
1: $M \leftarrow (X_1G_K)^\top - I_{2^n}$;
2: if $\mathrm{Row}_r(M) = \mathbf{0}_{2^n}^\top$ then
3:     Construct $\widetilde{M}$ by remove $\mathrm{Row}_r(M)$ and $\mathrm{Col}_r(M)$;
4: else
5:     Construct $\widetilde{M}$ by remove $\mathrm{Col}_r(M)$;
6: end if
7: $\min \mathbf{1}_{2^n-1}$ s.t. $\begin{cases} \widetilde{M}\tilde{q} < 0, \\ \tilde{q} > 0; \end{cases}$
8: if $\tilde{q}$ is found then
9:     $q \leftarrow [\tilde{q}_1, \ldots, \tilde{q}_{r-1}, 0, \tilde{q}_{r+1}, \ldots, \tilde{q}_{2^n-1}]$;
10: else
11:     $q = []$;
12: end if
Output: $q$.
```

Proof. This proposition follows directly from Theorem 3, and the proof is therefore omitted.

In the general case where no controller is given in advance, the task is to determine whether the system is stabilizable and, if so, to construct an appropriate controller. A natural approach is to search for a real matrix G_K satisfying Theorem 2, but this is generally difficult due to the large solution space. To simplify this problem, we instead work with logical matrices, as shown below.

Lemma 4. Under Assumption 1, BCN (4) is finite-time $\delta_{2^n}^r$-stabilizable via the feedback law (14) if and only if there exist $q \in \mathbb{R}^{2^n}$ and $G_K \in \mathcal{L}^{T\times 2^n}$ satisfying (13) and (15).

Proof. See Appendix A.

Building on this result, we propose a constructive approach for determining system stabilizability and designing a stabilizing controller.

Proposition 2. BCN (4) is finite-time x_e-stabilizable via state-feedback control if and only if Algorithm 2 yields a feasible K.

Proof. The proposition follows directly from Lemma 4.

Algorithm 2 Design state-feedback controllers for a BCN.

```
Input: $U_0$, $X_0$, $X_1$, $r$ and iteration times $T$.
Initialize: $G_K = \mathbf{0}_{T\times 2^n}$, and empty lists $\mathbb{G}$, $\mathbb{K}$.
1: Initialize $G_K(\arg\max_i \|X_1(i,\mathrm{col})\|_2, \mathrm{col}) \leftarrow 1$ for each col $\in [1:2^n]$;
2: for $t = 1:T$ do
3:     $F \leftarrow X_1(U_0 * X_0)^\dagger D_{[2^m,2^n]}(U_0 * X_0)G_K\Phi_n$;
4:     Obtain $q$ by Algorithm 1;
5:     if $q$ is valid then
6:         $K \leftarrow D_{[2^m,2^n]}(U_0 * X_0)G_K$;
7:         if $K$ is distinct then
8:             Append $G_K$, $K$ to $\mathbb{G}$, $\mathbb{K}$;
9:         end if
10:     end if
11:     Randomly modify one column of $G_K$;
12: end for
Output: Sets of feasible $\mathbb{G}$ and $\mathbb{K}$.
```

The following examples illustrate Algorithms 1 and 2.

Example 1. Consider BCN (4) with $n = 2$ and $m = 1$. We aim to determine whether it is stabilized at $x_e = \delta_4^1$ under given state-feedback controllers.

The data are collected by applying input sequences designed according to Procedure 2 in Subsection 4.3, and are given as follows:

$$U_0^1 = \delta_2[1,1],\ U_0^2 = \delta_2[1,1,1],\ U_0^3 = \delta_2[2,2,2,2],$$
$$X_0^1 = \delta_4[1,1],\ X_0^2 = \delta_4[2,4,3],\ X_0^3 = \delta_4[2,1,4,3],$$
$$X_1^1 = \delta_4[1,1],\ X_1^2 = \delta_4[4,3,2],\ X_1^1 = \delta_4[1,4,3,4].$$

Consider two candidate gain matrices, $K_1 = \delta_2[1,2,1,2]$ and $K_2 = \delta_2[1,1,2,2]$. Their corresponding state transition graphs are shown in Figure 1.

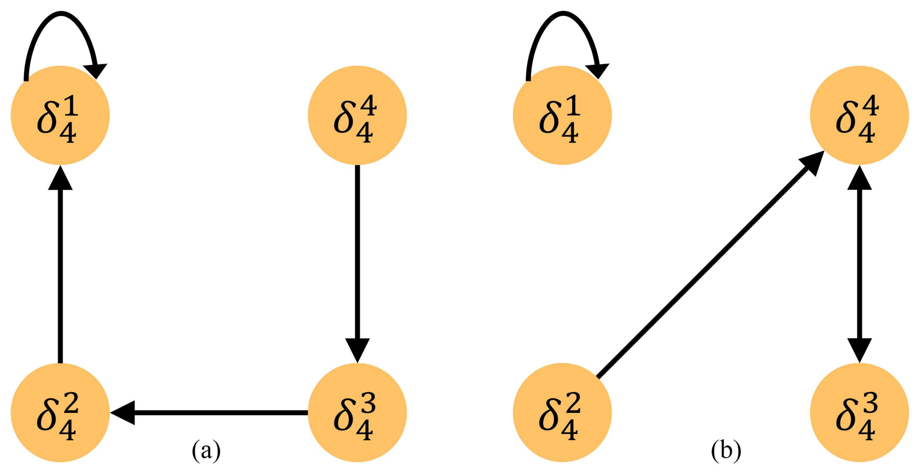

Figure 1 (Color online) State transition graphs under gain matrices (a) K_1 and (b) K_2, respectively.

Using Algorithm 1, we verify that (13) has feasible solutions for K_1, and one such solution is $q = [0, 1, 2, 3]^\top$. Therefore, BCN (4) is finite-time δ_4^1-stabilizable under the control law $u(t) = K_1x(t)$. In contrast, no feasible solution exists for K_2, which implies that the system cannot achieve stabilization under $u(t) = K_2x(t)$.

Example 2. Consider a reachable BCN (4) with $n = 3$ and $m = 1$. The data are collected by designing an input sequence as in Procedure 1 over the interval $[0 : 18]$, and are rearranged as

$$\begin{aligned}
U_0 &= \delta_2[1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 1],\\
X_0 &= \delta_8[6, 7, 6, 7, 1, 2, 1, 2, 3, 4, 3, 4, 5, 4, 5, 8, 1, 8],\\
X_1 &= \delta_8[7, 6, 7, 1, 2, 1, 2, 3, 4, 3, 4, 5, 4, 5, 8, 1, 8, 8].
\end{aligned}$$

A simple rank check confirms that Assumption 1 holds. Algorithm 2 is then applied to judge the stabilizability at $x_e = \delta_8^8$. As a result, all 12 distinct stabilizing K are found, which can be expressed as

$$K = \delta_2[i_1, i_2, k, 2, 1, k, 2, 1],$$

where $(i_1, i_2) \in \{(1, 2), (2, 1), (2, 2)\}$, and $k \in \{1, 2\}$.

4.2 Event-triggered control from data

In this subsection, we show that the previously designed state-feedback controller can also be implemented under an event-triggered mechanism. Specifically, the control input is updated only when necessary rather than continuously. The event-triggered control (ETC) takes the following form:

$$u(t) = Kx(t_k),\ t \in [t_k, t_{k+1}), \tag{18}$$

where the sampling instants $\{t_k \mid t \in \mathbb{N}\}$ are determined by a triggering policy. Without loss of generality, we assume that $t_0 = 0$.

Assumption 2 is adopted throughout this section.

Assumption 2. BCN (4) is finite-time x_e-stabilizable.

We begin by presenting the data-based representation of BCN (4) under ETC (18).

Theorem 4. Let Assumptions 1 and 2 hold. Then, the closed-loop BCN (4) under ETC (18) has the following equivalent representation:

$$x(t) = (X_1\widetilde{G}_K)^{t-t_k}\Phi_n^{t-t_k}x(t_k),\ t \in [t_k, t_{k+1}), \tag{19}$$

where $\widetilde{G}_K \in \mathbb{R}^{T\times 2^{2n}}$ satisfies

$$(U_0 * X_0)\widetilde{G}_K = KW_{[2^n,2^n]}I_{2^{2n}}, \tag{20}$$

and $W_{[2^n,2^n]} = [I_{2^n} \otimes \delta_{2^n}^1, I_{2^n} \otimes \delta_{2^n}^2, \ldots, I_{2^n} \otimes \delta_{2^n}^{2^n}]$ is a swap matrix. Here, $I_{2^{2n}}$ is included to ensure dimension compatibility under the semitensor product.

Proof. See Appendix B.

Table 1 Sampling scheduling.

x_0	Sampling sequence	x_0	Sampling sequence
δ_8^1	$\{0,1\}$	δ_8^5	$\{0\}$
δ_8^2	$\{0,3\}$	δ_8^6	$\{0,1,3\}$
δ_8^3	$\{0,1,2\}$	δ_8^7	$\{0,2\}$
δ_8^4	$\{0,1\}$	δ_8^8	$\{0\}$

When Assumption 2 holds, a feasible vector $\tilde{q} \in \mathbb{R}^{2^n}$ (satisfying (13)) and a matrix $\widetilde{G}_K \in \mathbb{R}^{T \times 2^{2n}}$ (satisfying (20)) can be constructed. The event-triggered mechanism is defined as $t_0 = 0$ and

$$t_{k+1} = \begin{cases} \inf\left\{t > t_k \,\middle|\, x^\top(t_k)F(t,t_k)^\top \tilde{q} \geqslant 0\right\}, & \text{if } x(t) \neq x_e, \\ +\infty, & \text{otherwise}, \end{cases} \tag{21}$$

where $F(t,t_k) = (X_1\widetilde{G}_K)^{t-t_k}\Phi_n^{t-t_k}(X_1\widetilde{G}_K\Phi_n - I)$. This mechanism guarantees, by design, that $V(x(t+1)) - V(x(t)) < 0$ holds along the trajectories of (19).

Theorem 5. Suppose Assumptions 1 and 2 hold. For BCN (4) with a state-feedback gain $K = D_{[2^m,2^n]}(U_0 * X_0)G_K$, where G_K is any solution to (13), the following statements hold.

(1) A global maximum number of triggers exists. In particular, for any initial state $x_0 \in \Delta_{2^n}$, there exists an integer $N < 2^n$ such that for any $t \geqslant t_N$, the control input $u(t)$ remains constant; that is, $t_N < +\infty$ and $t_{N+1} = +\infty$.

(2) BCN (4) is finite-time x_e-stabilizable under ETC (18), where triggering instants are determined by (21).

Proof. When Assumption 2 holds, according to the Lyapunov function defined in (16), we have $V(x(t+1)) < V(x(t))$ if $x(t) \neq x_e$, and $V(x(t+k)) = V(x(t)) = 0$ for all $k \in \mathbb{N}$ when $x(t) = x_e$. Since the state space Δ_{2^n} is finite, $V(x(t))$ can take at most 2^n distinct values. Intuitively, this means that the system can only be triggered a finite number of times before reaching the target state.

We first prove statement (1) by contradiction. Suppose $N \geqslant 2^n$, then there must exist some $\tau \in [0, N)$ such that $V(x(t_\tau)) = 0$, which implies that $x(t_\tau) = x_e$. According to the control law $u(t) = Kx(t_\tau),\ t \geqslant t_\tau$, and given that BCN (4) is finite-time x_e-stabilizable, the state remains at x_e for all $t \geqslant t_\tau$. As a result, $V(x(t)) = 0,\ t \geqslant t_\tau$. By event-triggered mechanism (21), no further control updates will occur, i.e., $t_{\tau+1} = +\infty$, contradicting the assumption that $t_N = +\infty$. Hence, we must have $N < 2^n$.

Next, we prove statement (2). By event-triggered mechanism (21), the Lyapunov function is strictly decreasing along the triggering sequence:

$$V(x(t_0)) > V(x(t_1)) > \cdots > V(x(t_N)) \geqslant V(x(t_{N+1})) = 0.$$

If $x(t_N) = x_e$, then controller $u(t) = Kx(t_N)$ ensures that the system stays at x_e afterwards, and thus achieves x_e-stabilization in finite time t_N. If $x(t_N) \neq x_e$, controller $u(t) = Kx(t_N)$ still drives the system to x_e in at most 2^n steps due to the finiteness of the state space. Therefore, stabilization at x_e is guaranteed no later than $t_N + 2^n$. This completes the proof.

Example 3. We now revisit Example 2. A valid gain matrix is expressed as $K = \delta_2[2,2,1,2,1,1,2,1]$. Based on Algorithm 1, a corresponding Lyapunov function is given by

$$V(x(t)) = x^\top(t)q,$$

where $q = [1,4,3,2,1,3,2,0]^\top$.

In this setup, the event-triggered mechanism (21) can be applied. The resulting triggering sequence $\{t_k \mid k \in \mathbb{N}\}$ for different initial states $x(0) \in \Delta_8$ is computed accordingly in Table 1.

4.3 Guideline for data collection

This subsection presents guidelines for collecting input-state data that satisfy Assumption 1 by designing suitable input sequences. Depending on the experimental setting, consider two representative cases.

Case 1: single-experiment setting. In scenarios where only one continuous experiment can be performed, Assumption 3 is essential. It guarantees that all state-input pairs can be visited by a single input sequence.

Assumption 3. BCN (4) is reachable; that is, for any $x_0 \in \Delta_{2^n}$, there exists a control sequence $\mathbf{u}$ that drives x_0 to any $x^* \in \Delta_{2^n}$.

Without loss of generality, we initialize the system at $x_0 = \delta_{2^n}^1$. Procedure 1 shows how to design a feasible control sequence.

Procedure 1 (Design of input sequences in a single-experiment)**.**

• **Stage 1: fixed input exploration.** For each fixed input $u = \delta_{2^m}^k, k \in [1 : 2^m]$, proceed with

(1) Initialize $x_0^1 = \delta_{2^n}^1$. For $k \geqslant 2$, set $x_0^k = x_{T_{k-1}-1}^{k-1}$.

(2) Apply $u = \delta_{2^m}^k$ and record the resulting trajectory $\{x_t^k\}_{t=0}^{T_k-1}$ until a state repeats at time T_k.

The input sequence generated in this stage is

$$\mathbf{u}_1 = \bigcup_{k=1}^{2^m} \underbrace{\{\delta_{2^m}^k, \ldots, \delta_{2^m}^k\}}_{T_k \text{ times}}.$$

• **Stage 2: completion of unexplored transitions.** Define the set of unexplored transitions

$$\mathcal{E} := \{(x, u) \mid x \in \mathcal{X}_k := \Delta_{2^n} \backslash \{x_t^k\}_{t=0}^{T_k-1},\ u = \delta_{2^m}^k,\ k \in [1 : 2^m]\}.$$

Initialize $\mathbf{u}_2 = [\,]$. While $\mathcal{E} \neq \emptyset$, repeat the following procedures.

(1) Select a transition $(\tilde{x}, \tilde{u}) \in \mathcal{E}$: identify a shortest control sequence $\mathbf{v}_{\tilde{x}}$ that drives the current state to $\tilde{x}$. Apply the input $\tilde{u}$ to transition to a new state $\tilde{x}'$. Remove $(\tilde{x}, \tilde{u})$ from $\mathcal{E}$ and update the sequence as $\mathbf{u}_2 \leftarrow \mathbf{u}_2 \cup \mathbf{v}_{\tilde{x}} \cup \{\tilde{u}\}$.

(2) While there exists an input $\tilde{u}'$ such that $(\tilde{x}', \tilde{u}') \in \mathcal{E}$: apply $\tilde{u}'$ to transition from $\tilde{x}'$ to a new state $\tilde{x}''$. Then, remove $(\tilde{x}', \tilde{u}')$ from $\mathcal{E}$, and update $\tilde{x}' \leftarrow \tilde{x}''$ and $\mathbf{u}_2 \leftarrow \mathbf{u}_2 \cup \{\tilde{u}'\}$.

• **Output.** The complete control input sequence $\mathbf{u} = \mathbf{u}_1 \cup \mathbf{u}_2$.

Theorem 6. Suppose Assumption 3 holds. The control sequence generated by Procedure 1 ensures that condition (8) is satisfied.

Proof. See Appendix C.

Case 2: multi-experiment setting. Consider that multiple experiments can be conducted, possibly with different initial conditions. In this case, reachability is no longer required, and the design procedure is described as follows.

Procedure 2 (Design of input sequences in multi-experiments)**.**

• **Stage 1: initial experiment.** Apply Procedure 1 in a single experiment. Terminate once no further transitions in $\mathcal{E}$ can be realized.

• **Stage 2: additional experiments.** While $\mathcal{E} \neq \emptyset$, select a pair $(x, u) \in \mathcal{E}$ as the initial condition for a new experiment. Repeat Stages 1 to obtain a new input sequence $\mathbf{u}^i$, where i is the experiment index. After each attempt, update $\mathcal{E}$ by removing all pair that have been visited. If the resulting trajectory fails to realize any new transition in $\mathcal{E}$, terminate the experiment and select another $(x, u) \in \mathcal{E}$ for the next run.

• **Output.** A collection of input sequences $\{\mathbf{u}^i\}_{i \in \mathbb{N}}$.

Let q experiments be performed. The resulting data are arranged in the form (7) as follows:

$$U_{[0:T-1]} = [U_{[0:T_1-1]}^1, \ldots, U_{[0:T_q-1]}^q], \quad X_{[0:T-1]} = [X_{[0:T_1-1]}^1, \ldots, X_{[0:T_q-1]}^q],$$

where $T = \sum_{i=1}^q T_i$, and $U_{[0:T_i-1]}^i, X_{[0:T_i-1]}^i$ are collected from the ith experiment. Then we have Proposition 3.

Proposition 3. In the multi-experiment setting, the control sequences generated by Procedure 2 ensure condition (8).

Proof. Since any unreachable state can be directly selected as the initial condition of a new experiment, all state-input pairs can eventually be explored. The result follows directly.

To illustrate the two data-collection procedures, we present Example 4.

Example 4. Consider two BCNs whose state transition graphs are shown in Figure 2. In practice, these graphs are unknown, and only input-state data are available; here, they are displayed solely to demonstrate how to collect data.

Case 1: reachable BCN (Procedure 1). For the BCN in Figure 2(a), in Stage 1, we get a trajectory $\delta_4^1 \xrightarrow{\delta_2^1} \delta_4^1 \xrightarrow{\delta_2^2} \delta_4^4 \xrightarrow{\delta_2^2} \delta_4^3 \xrightarrow{\delta_2^2} \delta_4^4$, which gives $\mathbf{u_1} = [\delta_2^1, \delta_2^2, \delta_2^2, \delta_2^2]$. In Stage 2, we first get the set of unexplored transitions $\mathcal{E} = \big\{(\delta_4^1, \delta_2^1), (\delta_4^2, \delta_2^1), (\delta_4^3, \delta_2^1), (\delta_4^2, \delta_2^2)\big\}$. To visit them, we construct the following path $\delta_4^4 \xrightarrow{\delta_2^1} \delta_4^3 \xrightarrow{\delta_2^1} \delta_4^2 \xrightarrow{\delta_2^1} \delta_4^4 \xrightarrow{\delta_2^1} \delta_4^3 \xrightarrow{\delta_2^1} \delta_4^2 \xrightarrow{\delta_2^2} \delta_4^4$, which gives $\mathbf{u_2} = [\delta_2^1, \delta_2^1, \delta_2^1, \delta_2^1, \delta_2^1, \delta_2^2]$. After this, all state-input pairs are covered. The complete dataset is

$$U_{[0:10]} = \delta_2[1, 2, 2, 2, 1, 1, 1, 1, 1, 2],\ X_{[0:11]} = \delta_4[1, 1, 4, 3, 4, 3, 2, 4, 3, 2, 4].$$

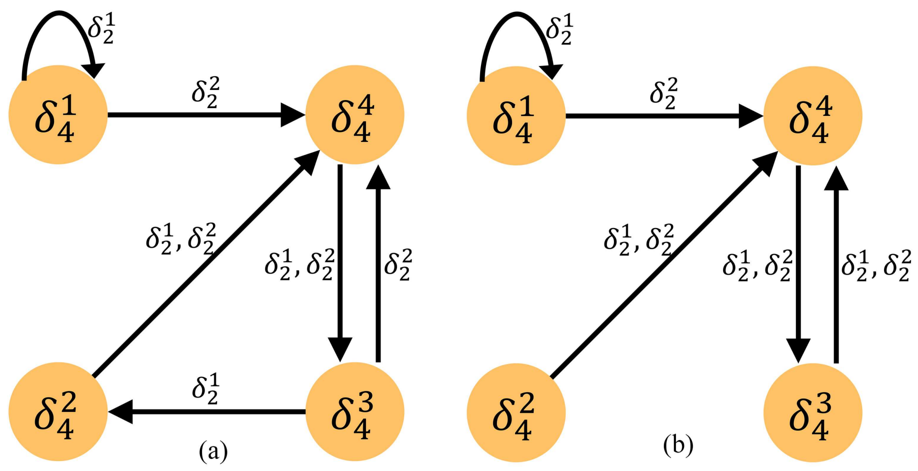

Figure 2 (Color online) State transition graphs of two BCNs. (a) A reachable BCN; (b) an unreachable BCN.

Case 2: unreachable BCN (Procedure 2). For the BCN in Figure 2(b), starting from δ_4^1, we obtain a trajectory $\delta_4^1 \xrightarrow{\delta_2^1} \delta_4^1 \xrightarrow{\delta_2^2} \delta_4^4 \xrightarrow{\delta_2^2} \delta_4^3 \xrightarrow{\delta_2^2} \delta_4^4 \xrightarrow{\delta_2^1} \delta_4^3 \xrightarrow{\delta_2^1} \delta_4^4$. The set of unvisited transitions is $\mathcal{E} = \{(\delta_4^2, \delta_2^1), (\delta_4^2, \delta_2^2)\}$, and in Stage 2, we perform two additional independent experiments: $\delta_4^2 \xrightarrow{\delta_2^1} \delta_4^4$ and $\delta_4^2 \xrightarrow{\delta_2^2} \delta_4^4$. The overall data are given as

$$U_{[0:6]}^1 = \delta_2[1, 2, 2, 2, 1, 1],\ U_{[0]}^2 = \delta_2[1],\ U_{[0]}^3 = \delta_2[2];$$
$$X_{[0:7]}^1 = \delta_4[1, 1, 4, 3, 4, 3, 4],\ X_{[0:1]}^2 = \delta_4[2, 4],\ X_{[0:1]}^3 = \delta_4[2, 4].$$

Remark 4. These two procedures correspond to fundamentally different experimental conditions. Procedure 1 applies when only one continuous experiment is feasible, where Assumption 3 is necessary to ensure data richness. A typical example is personalized medical therapy, where treatment data must be collected from a single patient without restarting [28]. By contrast, Procedure 2 applies when multiple independent experiments are possible, such as in multi-agent simulations [29], and reachability is not required.

Remark 5. Unlike linear systems, where persistently exciting inputs ensure data richness [24], BNs require explicitly designed input sequences to visit all relevant state-action pairs. In addition, while previous work (e.g., [20, 21]) focuses on designing a controller based on informative data, our framework guides the acquisition of such data, making it particularly suitable for multi-objective and constrained control problems.

5 Stabilizing BCNs under constrained control

This section extends the proposed data-driven framework to constrained control problems, where controllers must satisfy additional requirements while preserving stabilization. The following subsections present two representative scenarios to illustrate its applicability. Specifically, rather than enumerating all stabilizing controllers, the presented methods efficiently generate feasible ones by incorporating the constraints directly into the design process.

5.1 Control under temporal tasks

In many practical systems, control objectives extend beyond stabilization. For example, unmanned aerial vehicles may be required to visit several regions in a prescribed order before reaching the target position while satisfying energy, safety, and timing constraints [23, 30].

In BNs, such multi-task requirements can be reformulated as finite-time dwell problems [31]. Let $\{P_d(T_d^1, T_d^2, \tau_d) \mid d = 1, \dots, D\}$ denote the tasks, where T_d^1, T_d^2 are the earliest and latest completion time for the dth task, and τ_d is the dwell time. Define $T_{\max} = \max_d T_d^2$.

Under these tasks, the feasible choices of $G_K = \delta_T[g_1, g_2, \dots, g_{2^n}]$ are restricted. Algorithm 3 addresses this issue. Following that, when designing feasible controllers, G_K is initialized and updated with $\mathrm{Col}_i(G_K) \in \mathrm{candidate}(g_i), i \in [1 : 2^n]$ (i.e., modifying lines 1 and 11 in Algorithm 2), ensuring the constructed controllers meet all constraints.

Remark 6. Algorithm 3 is proposed to impose constraints on each column of G_K by identifying feasible control inputs. By these constraints, all temporal tasks can be satisfied while achieving finite-time stabilization.

Algorithm 3 Constraints on G_K.

```
Input: $X_1$, and $\{P_d(T_d^1, T_d^2, \tau_d), d = 1, \ldots, D\}$.
 1: Calculate the admissible state sets $A_t$ for all $t \in [0 : T_{\max}]$, and set $R_0 \leftarrow A_0$;    ▷ Get feasible/reachable state sets
 2: for $t = 0 : T_{\max} - 1$ do
 3:     Initialize $R_{t+1} \leftarrow \emptyset$, $\mathcal{C}_t(g_i) = [1 : T]$ for all $i \in [1 : 2^n]$, and update set $\mathcal{C}_t(g_i) = \{g_i \in [1 : T] \mid X_1\delta_T^{g_i} \in A_{t+1}\}$ for each $\delta_{2^n}^i \in R_t$;
                                    ▷ Feasible controls for reachable state $\delta_{2^n}^i$ such that the next state is admissible
 4:     for $\delta_{2^n}^i \in R_t$ do
 5:         if $\mathcal{C}_t(g_i) = \emptyset$ then
 6:             Return infeasible;    ▷ State $\delta_{2^n}^i$ has no feasible control
 7:         end if
 8:         for $k \in \mathcal{C}_t(g_i)$ do
 9:             $R_{t+1} = R_{t+1} \cup \{X_1\delta_T^k\}$;    ▷ Next reachable set after control
10:         end for
11:     end for
12: end for
13: for $\delta_{2^n}^i \in \bigcup_t R_t$ do
14:     candidate$(g_i) = \bigcap_{\{l \mid \delta_{2^n}^i \in R_l\}} \mathcal{C}_l(g_i)$;    ▷ Controls feasible at all times
15:     if candidate$(g_i) = \emptyset$ then
16:         Return infeasible;
17:     end if
18: end for
Output: Sets of feasible candidate$(g_i), i \in [1 : 2^n]$.
```

5.2 Optimal control

Performance is another critical aspect in controller design. Depending on the application, criteria such as minimum cost or fastest convergence may be prioritized. Here, we consider the problem of finding an optimal state-feedback controller that minimizes a cost function while maintaining stabilization.

For a BCN (4) with initial state x_0 and a control sequence $\mathbf{u} = \{u(t), t \in \mathbb{N}\}$, consider infinite-horizon cost function as

$$J(x_0, \mathbf{u}) = \sum_{t=0}^{\infty} h(x(t), u(t)),$$

where $h : \Delta_{2^n} \times \Delta_{2^m} \mapsto \mathbb{R}$ is the stage cost. Specifically, $h(x, u) = x^\top H u$, where $H_{i,j}$ denotes the cost of applying $u = \delta_{2^m}^j$ at state $x = \delta_{2^n}^i$.

By Theorem 3, the optimal control problem becomes:

$$\text{minimize } J(G_K) = \sum_{t=0}^{\infty} x^\top(t)\widetilde{H}G_K x(t) = \sum_{t=0}^{\infty} x^\top(t)\widetilde{H}\delta_T^{g_{\mathrm{idx}(x(t))}},$$

where $\widetilde{H} = HD_{[2^m,2^n]}(U_0 * X_0)$, and idx$(x(t))$ is the index of $x(t)$.

The optimal controller is designed via a Q-learning algorithm, given in Algorithm 4.

Algorithm 4 Optimal state-feedback control.

```
Input: iteration times $T$, $\alpha$, $\gamma$, $\epsilon$, $M \gg 0$.
 1: Initialize $Q(x, a) \leftarrow \mathbf{0}_{2^n \times T}$ and $G_K$ as in Algorithm 2;
 2: for $t = 1 : T$ do
 3:     $x \leftarrow \delta_{2^n}^i$, $i \leftarrow$ rand$(1 : 2^n)$;
 4:     for step $= 1 : 2^n$ do
 5:         Select an action $a$ according to $\epsilon$-greedy policy based on $Q(x, \cdot)$;
 6:         Execute action $a$, update $G_K$ by setting $g_i \leftarrow a$, and get the next state $x' = X_1\delta_T^a$;
 7:         $F \leftarrow X_1 G_K$ and obtain $q$ by Algorithm 1;
 8:         if $q$ is valid then
 9:             Compute per-step reward $r = -(\delta_{2^n}^i)^\top \widetilde{H}\delta_T^a$;
10:         else
11:             Set $r \leftarrow -M$;
12:             Break;
13:         end if
14:         Update $Q(x, a) \leftarrow (1 - \alpha)Q(x, a) + \alpha[r + \gamma \max_{a'} Q(x', a')]$;
15:         Set $x \leftarrow x'$;
16:     end for
17: end for
18: Obtain the optimal policy $\pi^*(x) = \arg\max_a Q(x, a)$;
19: Obtain $G_K^* = \delta_T[g_1, \ldots, g_{2^n}]$, $g_i = \pi^*(i)$;
Output: Optimal gain $K^* = D_{[2^m,2^n]}(U_0 * X_0)G_K^*$.
```

Table 2 Sampling scheduling under different x_0.

$(x_0, V(x_0))$	$\{t_k \mid k \in \mathbb{N}\}$, T	$(x_0, V(x_0))$	$\{t_k \mid k \in \mathbb{N}\}$, T
$(\delta_{256}^{22}, 8)$	$\{0, 2\}$, 5	$(\delta_{256}^{17}, 4)$	$\{0, 1\}$, 4
$(\delta_{256}^{1}, 7)$	$\{0, 2, 3\}$, 6	$(\delta_{256}^{6}, 3)$	$\{0, 1\}$, 3
$(\delta_{256}^{4}, 6)$	$\{0, 2\}$, 5	$(\delta_{256}^{5}, 2)$	$\{0\}$, 2
$(\delta_{256}^{25}, 5)$	$\{0\}$, 3	$(\delta_{256}^{256}, 1)$	$\{0\}$, 1

Its convergence is established by Lemma 5.

Lemma 5 (c.f. [32]). Assume BCN (4) is finite-time $\delta_{2^n}^r$-stabilizable. Then, under the following conditions:
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$;
- $\mathrm{var}[r_t]$ is finite.

Algorithm 4 converges to the fixed point $Q^*(x(t), u(t), \delta_{2^n}^r)$ for all $x(t) \in \Delta_{2^n}$ and $u(t) \in \Delta_{2^m}$ with probability one.

6 Simulations

Our proposed data-driven framework is well-suited for practical control problems where the system structure is unknown or difficult to model, particularly when the dynamics follow logical rules. Typical application areas include medical therapy [33] and gene regulation [34]. For example, this approach can be used to regulate cell concentrations [22], control gene activation or inhibition [34], or design personalized treatment strategies for diseases [28]. In these settings, although the exact logical functions are not known, the control goal can still be achieved using only the observed state-input data.

In this section, we use an *Escherichia coli* example to demonstrate it. Consider a reduced model of the *lac* operon in *Escherichia coli*, consisting of $n = 8$ state nodes and $m = 3$ controller nodes. This system is modeled by BCN (1), with the combined state and input variables defined as follows:

$$X = (A, M, A_m, L, C, R, L_m, R_m)^\top,\ U = (L_{em}, L_e, G_e)^\top.$$

In the model, the controller nodes L and G represent extracellular lactose and glucose, respectively; the state nodes A, M, L, C, R represent the allolactose, *lac* mRNA, lactose, CAP, LacI, and the subscripts m and e indicate medium and extracellular concentrations, respectively.

The logical functions $f_i, i \in [1:8]$ are assumed to be unknown. Our goal is to determine whether the system is stabilizable based on experimental data. If so, we design a suitable state-feedback controller and deal with constrained control problems.

The dataset is generated using input sequences designed according to Procedure 2 in Subsection 4.3, and the rank condition (8) is satisfied. We now illustrate our theoretical results in the following cases.

• Determination of stabilizability. We first check the stabilizability of BCN (1) at $X_e = (0, 0, 0, 1, 1, 1, 0, 0)^\top$ under two constant inputs $U \equiv (0, 0, 0)^\top$ and $U \equiv (1, 0, 0)^\top$. By Lemma 1, this is equivalent to analyzing the stability of BCN (4) at $x_e = \delta_{256}^{228}$ under $u \equiv \delta_8^8$ and $u \equiv \delta_8^4$, respectively.

Using Algorithm 1, a feasible solution $q \in \mathbb{R}^{256}$ is found in both cases, indicating that BCN (4) is finite-time x_e-stabilizable under the two inputs. This result is consistent with that in [35].

• Design of state-feedback controllers. Next, we consider a target state $x_e = \delta_{256}^{255}$. We aim to determine whether BCN (4) is x_e-stabilizable under state-feedback control.

To this end, we apply Algorithm 2, which yields several feasible gain matrices K. One such K, along with its corresponding q, is shown in Figure 3(a). Under this controller, the system achieves finite-time x_e-stabilization, as shown in Figure 3(b).

• Sampling schedule for ETC. By Theorem 5, the system can be finite-time stabilized at x_e by ETC (18). To determine the sampling schedule, we arbitrarily select initial states $x_0 \in \Delta_{256}$ corresponding to different Lyapunov levels $V(x)$. The resulting sampling scheduling for various initial states is presented in Table 2, where the time to stabilize is denoted as T.

• Optimal control under temporal tasks. Consider an optimal control problem in which the desired trajectory is required to satisfy the following tasks:

(i) At time step 1 or 2, the states of all nodes are active;

(ii) Within time steps 2 to 4, there exists a period of at least two consecutive steps during which A and M are inactive;

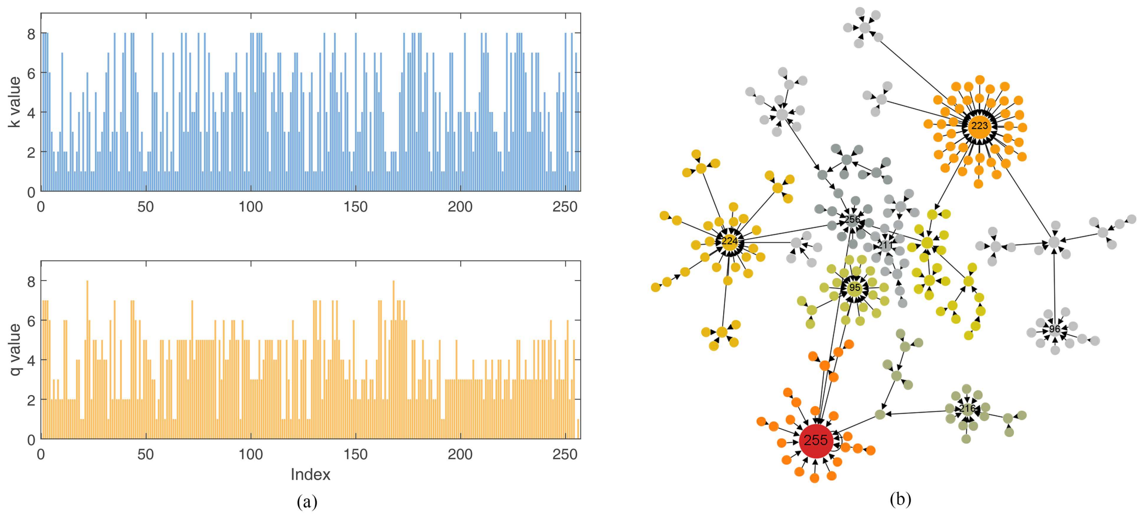

Figure 3 (Color online) (a) One feasible K and its corresponding q; (b) state transition graph under K, with x_e colored in red.

Table 3 Parameters of P_d.

P_d	Admissible states	T_d^1	T_d^2	τ_d
P_1	δ_{256}^1	1	2	1
P_2	δ_{256}^i, $i \in [193:256]$	2	4	2
P_3	δ_{256}^i, $i \in [65:192]$	5	6	1
P_4	δ_{256}^i, $i \in \{1,16,241,256\}$	7	7	1
P_5	δ_{256}^{255}	9	∞	∞

(iii) During time steps 5 to 6, there exists a step at which A and M take different values;
(iv) At time step 7, the nodes (A, M, A_m, L) and (C, R, L_m, R_m) each reach internal synchronization;
(v) The system is stabilized at x_e before time step 10.

The parameters of complex tasks P_d are summarized in Table 3.

The stage cost is given by

$$H_{i,j} = \begin{cases} j + 0.5\lceil \frac{i}{64} \rceil, & j \in [1:4], \\ 0.5j + \lceil \frac{i}{128} \rceil, & j \in [5:8], \end{cases}$$

where $i \in [1, 256]$ and $\lceil\cdot\rceil$ is the ceiling function.

To satisfy all $P_d, d \in [1:5]$, the initial state must be chosen from $\delta_{256}^i, i \in \{38, 40, 166, 168\}$. Then, according to Algorithms 3 and 4, the optimal gain matrix is obtained as $K^* = \delta_8[g_1^*, \ldots, g_{256}^*]$, with

$$g_i^* = \begin{cases} 2, \ i \in \{1, 38, 40, 69, 166, 168, 213, 214\}, \\ 1, \ \text{otherwise}. \end{cases}$$

7 Conclusion

This paper has presented a data-based framework for stabilizing BCNs under multiple control constraints, enabling the design to be directly derived from the observed data. To support implementation, practical procedures have been provided for obtaining sufficient data for analysis. The applicability of the framework has been illustrated through two constrained control problems, and its effectiveness has been further demonstrated using a biological example. A promising direction for future work is to extend the framework to scenarios with disturbed or missing data.

Acknowledgements This work was partially supported by Natural Science Foundation of Jiangsu Province (Grant Nos. BK20253024, BK20240009), National Natural Science Foundation of China (Grant No. 62373105), and Jiangsu Provincial Scientific Research Center of Applied Mathematics (Grant No. BK20233002).

References

1 Kauffman S A. Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol, 1969, 22: 437–467

2 Yao Y, Sun J, Zhang Y. Multitask synthesis of hybrid systems via temporal logic. IEEE Trans Automat Contr, 2023, 68: 6883–6890
3 Li C, Li A, Wu Y, et al. Logical dynamic games: models, equilibria, and potentials. IEEE Trans Automat Contr, 2024, 69: 7584–7599
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 Kitano H. Computational systems biology. Nature, 2002, 420: 206–210
6 Tang T Y, Lu J Q, Azuma S, et al. Polynomial-complexity distributed approaches for reachability of symmetric Boolean networks. IEEE Trans Automat Contr, 2025, doi: 10.1109/TAC.2025.3632662
7 Cheng D, Qi H, Li Z. Analysis and Control of Boolean Networks: A Semi-tensor Product Approach. Berlin: Springer, 2011
8 Li Z T, Guo Y Q, Gui W H. Asymptotical stability of continuous-time probabilistic logic networks based on transition rate. Sci China Inf Sci, 2023, 66: 132201
9 Yerudkar A, Del Vecchio C, Glielmo L. Feedback stabilization control design for switched Boolean control networks. Automatica, 2020, 116: 108934
10 Zhong D, Lu J, Liu Y, et al. Stabilizing logical networks with stochastic state-heritable control: a Markovian sampling method. SIAM J Control Optim, 2025, 63: 571–595
11 Zhu S, Lu J, Azuma S, et al. Strong structural controllability of Boolean networks: polynomial-time criteria, minimal node control, and distributed pinning strategies. IEEE Trans Automat Contr, 2023, 68: 5461–5476
12 Liu Y, Zhong J, Ho D W C, et al. Minimal observability of Boolean networks. Sci China Inf Sci, 2022, 65: 152203
13 Zhang Z, Zhang P, Leifeld T. Reduced-order observer design for fault diagnosis of Boolean control networks. Automatica, 2022, 146: 110618
14 Fornasini E, Valcher M E. Reconstructing the state of a Boolean control network via state feedback. IEEE Trans Automat Contr, 2023, 68: 5544–5551
15 Kobayashi K, Wu Y. Optimal reconstruction of noisy dynamics and selection probabilities in Boolean networks. Automatica, 2022, 136: 110094
16 Wang B, Feng J, Cheng D. On identification of Boolean control networks. SIAM J Control Optim, 2022, 60: 1591–1612
17 de Persis C, Postoyan R, Tesi P. Event-triggered control from data. IEEE Trans Automat Contr, 2024, 69: 3780–3795
18 de Persis C, Tesi P. Formulas for data-driven control: stabilization, optimality, and robustness. IEEE Trans Automat Contr, 2020, 65: 909–924
19 Wang Z M, Liu K Z, Cheng X L, et al. Online data-driven model predictive control for switched linear systems. IEEE Trans Automat Contr, 2025, 70: 6222–6229
20 Disarò G, Valcher M E. Data-based control of logical networks. Automatica, 2026, 187: 112888
21 Li R, Shi R, Zhang Q, et al. Data informativity for analysis and control design of Boolean control networks. SIAM J Control Optim, 2025, 63: 151–174
22 Montagna S, Braccini M, Roli A. The impact of self-loops on Boolean networks attractor landscape and implications for cell differentiation modelling. IEEE ACM Trans Comput Biol Bioinf, 2021, 18: 2702–2713
23 Xu J, Zeng Y, Zhang R. UAV-enabled wireless power transfer: trajectory design and energy optimization. IEEE Trans Wireless Commun, 2018, 17: 5092–5106
24 Willems J C, Rapisarda P, Markovsky I, et al. A note on persistency of excitation. Syst Control Lett, 2005, 54: 325–329
25 Khatri C G, Rao C R. Solutions to some functional equations and their applications to characterization of probability distributions. Sankhyā Indian J Stat Ser A, 1968, 30: 167–180
26 Li H, Wang Y. Lyapunov-based stability and construction of Lyapunov functions for Boolean networks. SIAM J Control Optim, 2017, 55: 3437–3457
27 Li R, Yang M, Chu T. State feedback stabilization for Boolean control networks. IEEE Trans Automat Contr, 2013, 58: 1853–1857
28 Goetz L H, Schork N J. Personalized medicine: motivation, challenges, and progress. Fertil Steril, 2018, 109: 952–963
29 Zhang H, Jiang H, Luo Y, et al. Data-driven optimal consensus control for discrete-time multi-agent systems with unknown dynamics using reinforcement learning method. IEEE Trans Ind Electron, 2017, 64: 4091–4100
30 Lindemann L, Dimarogonas D V. Feedback control strategies for multi-agent systems under a fragment of signal temporal logic tasks. Automatica, 2019, 106: 284–293
31 Yao Y, Sun J. Optimal control of multi-task Boolean control networks via temporal logic. Syst Control Lett, 2021, 156: 105007
32 Acernese A, Yerudkar A, Glielmo L, et al. Reinforcement learning approach to feedback stabilization problem of probabilistic Boolean control networks. IEEE Control Syst Lett, 2021, 5: 337–342
33 Tousley A M, Rotiroti M C, Labanieh L, et al. Co-opting signalling molecules enables logic-gated control of CAR T cells. Nature, 2023, 615: 507–516
34 Sgariglia D, Conforte A J, Pedreira C E, et al. Data-driven modeling of breast cancer tumors using Boolean networks. Front Big Data, 2021, 4: 656395
35 Veliz-Cuba A, Stigler B. Boolean models can explain bistability in the lac operon. J Comput Biol, 2011, 18: 783–794

Appendix A Proof of Lemma 4

[Sufficiency] The sufficiency is obvious by Theorem 3.

[Necessity] We show that if there exists a real matrix G_K satisfying conditions in Theorem 3, then there always exists a logical matrix $\widetilde{G}_K$ that satisfies the same conditions.

If Assumption 1 holds and $T = 2^n$, then G_K must be logical. For the case $T > 2^n$, define the index set

$$\mathscr{P}_i = \{j \in [1:T] \mid \mathrm{Col}_j(U_0 * X_0) = \delta^i_{2^{m+n}}\}.$$

Let $G_K = [g_{i,j}]$. Then the (i,j)th entry of $(U_0 * X_0)G_K$ becomes

$$[(U_0 * X_0)G_K]_{i,j} = \sum_{l=1}^{T} [U_0 * X_0]_{i,l}[G_K]_{l,j} = \sum_{l\in\mathscr{P}_i} g_{l,j}.$$

As a result, its jth column can be expressed as

$$\mathrm{Col}_j\left((U_0 * X_0)G_K\right) = \left[\sum_{l\in\mathscr{P}_1} g_{l,j}, \sum_{l\in\mathscr{P}_2} g_{l,j}, \ldots, \sum_{l\in\mathscr{P}_{2^{m+n}}} g_{l,j}\right]^\top,$$

where exactly one entry is 1, and all others are 0.

We now construct a logical matrix $\widetilde{G}_K = [\tilde{g}_{i,j}]$ as follows:

$$\tilde{g}_{\theta,j} = \begin{cases} \sum_{l \in \mathscr{P}_i} g_{l,j}, & \text{if}\, \theta = \min_w \{w \in \mathscr{P}_i\}, \\ 0, & \text{otherwise.} \end{cases}$$

This construction ensures that $\widetilde{G}_K$ is logical and satisfies

$$\mathrm{Col}_j((U_0 * X_0)G_K) = \mathrm{Col}_j((U_0 * X_0)\widetilde{G}_K).$$

Following a similar manner, we have $X_1 G_K = X_1 \widetilde{G}_K$. Therefore, replacing G_K with $\widetilde{G}_K$ preserves conditions (1) and (2).

Appendix B Proof of Theorem 4

When $t \in [t_k, t_{k+1})$, it can be derived by iteration that

$$\begin{aligned} x(t_k+1) &= Fu(t_k)x(t_k) = FKW_{[2^n,2^n]}\Phi_n x(t_k), \\ x(t_k+2) &= Fu(t_k)x(t_k+1) = (FKW_{[2^n,2^n]})^2\Phi_n^2 x(t_k), \\ &\vdots \\ x(t) &= (FKW_{[2^n,2^n]})^{t-t_k}\Phi_n^{t-t_k} x(t_k). \end{aligned}$$

According to Assumption 1, there exists a matrix $\widetilde{G}_K \in \mathbb{R}^{T \times 2^{2n}}$ such that

$$(U_0 * X_0)\widetilde{G}_K = KW_{[2^n,2^n]} I_{2^{2n}}.$$

In this way, Eq. (19) holds since $F(U_0 * X_0) = X_1$.

Appendix C Proof of Theorem 6

Following Procedure 1, the control sequence systematically explores all possible inputs in the input space Δ_{2^m}. To prove (8), it suffices to show that, for each input $u(t) \in \Delta_{2^m}$, the corresponding trajectories cover all states in Δ_{2^n}.

In Stage 1, for each fixed input $u = \delta_{2^m}^k$, the state trajectory visits all states in $\Delta_{2^n} \backslash \mathcal{X}_k$. Thus, transitions involving states in $\mathcal{X}_k$ must be visited in Stage 2. We consider the following two cases.

Case 1: $\bigcap_{k=1}^{2^m} \mathcal{X}_k = \emptyset$. Then every state appears in at least one trajectory in Stage 1. In Stage 2, we apply all remaining inputs at each of these states, ensuring all (x, u) pairs are covered.

Case 2: $\bigcap_{k=1}^{2^m} \mathcal{X}_k \neq \emptyset$. Let $\bigcap_{k=1}^{2^m} \mathcal{X}_k = \{x_0, x_1, \ldots, x_j\}$. By Assumption 3, there exists a control sequence that drives the system from x_0 to all x_j. Furthermore, a predecessor state $x_0^* \notin \bigcap_k \mathcal{X}_k$ and input u^* must exist such that $(x_0^*, u^*) \in \mathcal{E}$ and leads to x_0. (If no such x_0^* existed, x_0 would be unreachable from states outside the set, contradicting Assumption 3.) Therefore, by reaching x_0 from x_0^* and proceeding to all x_j, the entire set is visited.

In either case, all (x, u) pairs are covered, and the matrix $U_{[0:T-1]} * X_{[0:T-1]}$ contains all 2^{m+n} logical vectors, ensuring full row rank.