Article

State space analysis of diphtheria pathogenesis using semi-tensor products and permutation methods

Ugbene Ifeanyichukwu Jeff, **Ighovotueko Sophia Ajuremisan**
Department of Mathematics, Federal University of Petroleum Resources, Effurun, Nigeria
E-mail: ugbene.ifeanyi@fupre.edu.ng, ighovotuekosophia1@gmail.com

Received 9 March 2025; *Accepted 20 April 2025*; *Published online 20 May 2025*; *Published 1 December 2025*

Abstract
In this study, we analyze the state space of a Boolean network modeling diphtheria pathogenesis, focusing on key genes such as Tox, Rep, INF1/INF2, TLR, AP1, IL6, and TNF. We introduce targeted perturbations to reveal how the network responds and converges to its attractors. Our approach utilizes semi-tensor product techniques and permutation methods to recast the Boolean dynamics into a linear algebraic scheme, enabling efficient identification of transient states, stable attractors, and Garden-of-Eden states. This work fills an important gap by clarifying how specific gene interactions drive the network toward non-pathogenic states. Our results show that altering regulatory relationships, particularly those between Rep, Tox, and interferon signals, significantly influences basin sizes and attractor stability, thereby enhancing our understanding of the network's resilience and informing potential therapeutic strategies.

Keywords boolean network; state space analysis; perturbation; semi-tensor product; permutation method; diphtheria pathogenesis; attractor stability; Garden-of-Eden states; transient dynamics; gene regulation.

Network Biology
ISSN 2220-8879
URL: http://www.iaees.org/publications/journals/nb/online-version.asp
RSS: http://www.iaees.org/publications/journals/nb/rss.xml
E-mail: networkbiology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Diphtheria, a severe infectious disease caused by the bacterium *Corynebacterium diphtheriae*, has historically posed significant health challenges worldwide. Characterized by symptoms suchas sore throat, low-grade fever, and the formation of a pseudomembrane in the upper respiratorytract, diphtheria can lead to severe complications if not promptly treated. The primary virulencefactor of *C. diphtheriae* is the diphtheria toxin (Kolibo and Romaniuk, 2001), which inhibitsprotein synthesis in host cells, leading to tissue damage and systemic effects. Understanding thecomplex interactions among the genes involved in toxin production (Schmitt et al., 1992), immune response (Moehring et al., 1971), and disease transmission is crucial for developing effectivetherapeutic strategies (Harris et al., 2002; Kashiwagi et al., 2014).

In recent years, computational modeling has emerged as a powerful tool to dissect the intricate gene regulatory networks underlying various pathogenic processes (Qu et al., 2015; Possieri and Teel, 2017; Zhang, 2018; Menini et al., 2019; Shu et al., 2021; Kayoh and Ugbene, 2023; Ugbene and Utoyo-Ovokaeefe, 2024; Ugbene and Agwemuria, 2024), including diphtheria. One such approach is the use of Boolean networks, which provide a simplified yet insightful representation of geneinteractions by modeling genes as binary variables, either active (1) or inactive (0) (Wilson et al., 2019). This scheme allows researchers to simulate and analyze the dynamic behavior of genenetworks under different conditions. A notable study by Ugbene and Utoyo (2024) employed a Boolean network model to investigatethe gene regulatory mechanisms associated with diphtheria pathogenesis. Their model focusedon eight critical genes implicated in toxin production and immune response (Pimenta et al., 2008a; Holmes, 2000; Hadfield et al., 2000). Through dynamic simulations, they identified three distinct attractors within the network, each corresponding to a potential disease state (Belsey et al., 1969; Pimenta et al., 2008b). These attractors represent stable patterns of gene expression that thesystem gravitates toward over time, offering valuable insights into the possible outcomes of geneinteractions during infection.

While the identification of attractors provides essential information about the stable states ofthe system, a comprehensive understanding of the disease mechanism necessitates a thoroughexploration of the entire state space of the Boolean network. The state space encompasses all possible configurations of the network, detailing how the system transitions from one state to another (Yu et al., 2019; Yang et al., 2020; Yuan et al., 2019). Analyzing this space can reveal transient states, pathways leading to attractors, and potential intervention points to alter diseaseprogression. To achieve a detailed analysis of the state space, advanced mathematical tools and methodologies are required. One such approach is the semi-tensor product (STP) method developed by Cheng and Qi (2010). The STP technique transforms logical functions into algebraic forms, enabling the representation of Boolean networks as discrete-time linear dynamic systems. This algebraic representation facilitates the systematic examination of network properties, including fixed points, cycles, and transient behaviors (Wang and Albert, 2009; Wang et al., 2020; Veliz-Cubaet al., 2014). By converting the Boolean dynamics into a linear scheme, the STP method allowsor the application of linear system theories to analyze the network's structural and functional characteristics. Another innovative methodology is the permutation-based approach introduced by Wang et al. (2023). This technique involves representing the state transitions of a Boolean network using permutation matrices, which capture the rearrangement of states under the network's dynamics. Bystudying these permutations, researchers can gain insights into the cyclical patterns and invariant subsets within the state space. This method offers a unique perspective on the network's behavior, particularly in identifying symmetries and conserved structures that may not be apparent through other analytical techniques.

In this study, we aim to build upon the foundational work of Ugbene and Utoyo (2024) by conducting an in-depth analysis of the state space of the perturbed diphtheria pathogenesis Boolean network. Utilizing the semi-tensor product method of Cheng and Qi (2010) alongside the permutation-inspired approach of Wang et al. (2023), we seek to uncover the full spectrum ofdynamic behaviors exhibited by the network. Our objectives include identifying all possible statetransitions, mapping out the pathways leading to various attractors, and pinpointing potential control nodes that could serve as targets for therapeutic intervention. By integrating these advanced analytical methodologies, we strive to provide a comprehensive understanding of the gene regulatory mechanisms driving diphtheria pathogenesis. Such insights are essential for the development of targeted strategies aimed at disrupting the disease process and improving patient outcomes.

2 Methodology

In this section, we introduce a different method for analyzing Boolean networks.

2.1 Semi-tensor products

As described in Cheng and Qi (2010), consider a row vector U of dimension mq and a column vector V of dimension q. We partition U into q segments, denoted as $U^1, U^2, \dots, U^q$, each forming a $1 \times m$row. The semi-tensor product (STP), represented by $\ltimes$ (left semi tensor product), is defined as:

$$\begin{cases} U \ltimes V = \sum_{k=1}^{q} U^k v_k \in \mathbb{R}^m, \\ V^T \ltimes U^T = \sum_{k=1}^{q} v_k (U^k)^T \in \mathbb{R}^m. \end{cases} \quad (1)$$

Given matrices $P \in M_{r\times m}$ and $Q \in M_{s\times t}$, if m is a multiple of s, such that$m = sk$, we denote this as $P \succ_k Q$. Alternatively, if s is a multiple of m, so that $s = mk$, we denote it as $P \prec_k Q$. The semi-tensor product of P and Q, expressed as $D = P \ltimes Q$, consists of $r \times t$blocks structured as:

$$D^{ij} = P^i \ltimes Q_j, i = 1, \dots, r, j = 1, \dots, t, \quad (2)$$

where P^i represents the i-th row of P and Q_j denotes the j-th column of Q. Fundamental properties of the STP include: The STP adheres to the following rules (whenever the operations are well defined):

1. Distributive Property

$$P \ltimes (\lambda Q + \mu R) = \lambda P \ltimes Q + \mu P \ltimes R; (3)$$
$$(\lambda Q + \mu R) \ltimes P = \lambda Q \ltimes P + \mu R \ltimes P, \lambda, \mu \in \mathbb{R}. (4)$$

2. Associative Property

$$P \ltimes (Q \ltimes R) = (P \ltimes Q) \ltimes R. \quad (5)$$

If $P \succ_h Q$, then (where $\otimes$ represents the Kronecker product):

$$P \ltimes Q = P(Q \otimes I_h); \quad (6)$$

Likewise, if $P \prec_h Q$, then:

$$P \ltimes Q = (P \otimes I_h)Q. \quad (7)$$

Given $P \in M_{r\times m}$:

1. For a row vector$W \in \mathbb{R}^{1\times s}$:

$$P \ltimes W = W \ltimes (I_s \otimes P); \quad (8)$$

2. For a column vector $W \in \mathbb{R}^{s\times 1}$:

$$W \ltimes P = (I_s \otimes P) \ltimes W. \quad (9)$$

If $P \in M_{r\times m}$ and either r divides m or vice versa, then:

$$[P^h := \underbrace{P \ltimes \cdots \ltimes P}_{h\text{times}}]$$

is well-defined. Notably, for any column or row vector ζ, ζ^h is always valid.Define a delta set as $\Delta_h := \{\gamma_h^j | j = 1,2, \dots, h\}$, where γ_h^j represents the j-th column of the identity matrix I_h. A matrix $P \in M_{r\times m}$is classified as a logic matrix if $r = 2^a$ and $m = 2^b$ for some $a, b \in \mathbb{Z}_+$. The set of all such logic matrices is denoted as $\mathcal{L}$. A direct computation reveals, that:

If $P, Q \in \mathcal{L}$, then $P \ltimes Q \in \mathcal{L}$.

This property is useful in Boolean network models, where relevant matrices belong to $\mathcal{L}$, ensuring that their semi-tensor products are well-defined.For a matrix $P = [\gamma_r^{j_1}, \gamma_r^{j_2}, \dots, \gamma_r^{j_m}]$, we use the compact notation:

$$P = \gamma_r[j_1, j_2, \dots, j_m]$$

A swap matrix $S[r,m]$, an $rm \times rm$ matrix, is constructed by labeling its columns as $(11,12,\dots,1m,\dots,r1,r2,\dots,rm)$ and its rows as $(11,21,\dots,r1,\dots,1m,2m,\dots,rm)$. The elements at position $((K,L),(k,l))$ are given by:

$$s_{(K,L),(k,l)} = \gamma_{K,k}\gamma_{L,l} = \begin{cases} 1, & K = k \text{and} L = l, \\ 0, & \text{otherwise.} \end{cases}$$

For $r = m$, we use the notation $S_{[m]} := S[m,m]$. For example, setting $r = 3$ and $m = 2$, the swap matrix $S_{[3,2]}$ is:

$$S_{[3,2]} = \begin{bmatrix} (11) & (12) & (21) & (22) & (31) & (32) & \\ 1 & 0 & 0 & 0 & 0 & 0 & (11) \\ 0 & 0 & 0 & 1 & 0 & 0 & (22) \\ 0 & 1 & 0 & 0 & 0 & 0 & (12) \\ 0 & 0 & 0 & 0 & 1 & 0 & (31) \\ 0 & 0 & 1 & 0 & 0 & 0 & (21) \\ 0 & 0 & 0 & 0 & 0 & 1 & (32) \end{bmatrix}$$

which simplifies to:

$$S_{[3,2]} = \gamma_6[1,3,5,2,4,6].$$

For $U \in \mathbb{R}^r$ and$V \in \mathbb{R}^m$:

$$S_{[r,m]} \ltimes U \ltimes V = V \ltimes U, S_{[m,r]} \ltimes V \ltimes U = U \ltimes V.$$

If $P \in M_{r\times m}$ and $Q \in M_{m\times t}$, then:

$$PQ = P \ltimes Q.$$

This establishes the semi-tensor product as a generalization of conventional matrix multiplication, allowing omission of the $\ltimes$ symbol when standard matrix multiplication applies.

Now, we revisit the matrix representation of logical operations. Under this scheme, a logical variable is represented as a vector, while an $m-$ary logical function is represented by a 2×2^m matrix, referred to as the structural matrix of the function. This formulation allows logical functions acting on m logical variables to be expressed as a matrix-vector product. Further details can be found in Cheng and Qi (2005). First, we introduce some fundamental notations and results concerning logical expressions. The logical domain, denoted by E, is given by:

$$E = \{A = 1, B = 0\}.$$

An $m-$ary logical function is defined as a mapping $f: E^m \to E$. To adopt the matrix representation, we associate each element in E with a vector as $A \sim \begin{bmatrix}1\\0\end{bmatrix}^T$ and $B \sim \begin{bmatrix}0\\1\end{bmatrix}^T$, and define:

$$E_v = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\}.$$

Using this vector representation, we define the structural matrix of a logical function. A 2×2^m matrix N_ψ is termed the structural matrix of an $m-$ary logical function ψ, if

$$\psi(Q_1, Q_2, \dots, Q_m) = N_\psi Q_1 Q_2 \dots Q_m, \text{where } Q_1, \dots, Q_m \in E_v.$$

If such a matrix exists, it uniquely characterizes the logical function. To demonstrate the existence of this matrix for every logical function, we introduce a preparatory concept. Define a matrix, referred to as the power-reduction matrix, as follows (in condensed notation):

$$N_s = \gamma_4[1,4].$$

The name derives from the following property. Let $Q \in E_v$. Then we have

$$Q^2 = N_s Q. \tag{10}$$

A proof of the subsequent theorem can be found in (Qi and Cheng, 2008).Any logical function $G(Q_1, \dots, Q_m)$with logical arguments $Q_1, \dots, Q_m \in E_v$ can be expressed in the canonical

$$G(Q_1, \dots, Q_m) = N_G Q_1 Q_2 \dots Q_m, \tag{11}$$

where $N_G$is a 2×2^m matrix, known as the structural matrix of G. Next, consider a fundamental unary logical operation: Negation, $\neg Q$, along with four fundamental binary logical functions (Rade and Westergren, 1998): Disjunction, $Q_1 \vee Q_2$; Conjunction, $Q_1 \wedge Q_2$; Implication, $Q_1 \to Q_2$; Equivalence, $Q_1 \leftrightarrow Q_2$. Their structural matrices are as follows:

$$\begin{array}{ll} N_\neg := N_m & = \gamma_2[2,1]; \\ N_\vee := N_d & = \gamma_2[1,1,1,2]; N_\wedge = N_c := \gamma_2[1,2,2,2]; \\ N_\to = N_i & = \gamma_2[1,2,1,1]; N_\leftrightarrow := \gamma_2[1,2,2,1]. \end{array} \tag{12}$$

$$F(X,Y) = (X \longrightarrow Y) \vee (\neg X).$$

Using the vector representation of logical variables, equation (12), and the transformation matrix, we obtain:

$$\begin{aligned} F(X,Y) &= N_d(N_i XY)(N_m X) \\ &= N_d N_i (I_4 \otimes N_m) XYX \\ &= N_d N_i (I_4 \otimes N_m) X S_{[2]} XY \\ &= N_d N_i (I_4 \otimes N_m)(I_2 \otimes S_{[2]}) X^2 Y \\ &= N_d N_i (I_4 \otimes N_m)(I_2 \otimes S_{[2]}) N_s XY. \end{aligned}$$

Thus, we conclude that

$$N_F = N_d N_i (I_4 \otimes N_m)(I_2 \otimes S_{[2]}) N_s = \gamma_2[1,2,1,1].$$

2.2 Dynamics of Boolean networks

A Boolean network (Farrow et. al, 2004) consisting of nodes $B_1, B_2, \dots, B_m$ is defined as:

$$\begin{cases} B_{1\tau+1} & = g_1(B_{1\tau}, B_{2\tau}, \dots, B_{m\tau}), \\ B_{2\tau+1} & = g_2(B_{1\tau}, B_{2\tau}, \dots, B_{m\tau}), \\ \vdots & \\ B_{m\tau+1} & = g_m(B_{1\tau}, B_{2\tau}, \dots, B_{m\tau}), \end{cases} \tag{13}$$

where g_i, for $i = 1,2, \dots, m$, are Boolean functions.Consider a Boolean network given by:

$$\begin{cases} B_{\tau+1} & = C_\tau \wedge D_\tau, \\ C_{\tau+1} & = \neg B_\tau, \\ D_{\tau+1} & = C_\tau \vee D_\tau. \end{cases} \tag{14}$$

To express this system in algebraic form, we define:

$$z_\tau = \ltimes_{i=1}^{m} B_{i\tau} \quad (15)$$

The mapping $\ltimes_{i=1}^{m}: \Delta_2^m \to \Delta_{2^m}$ is bijective, allowing us to recover B_i from z using equation (22). Applying equation (11), we can derive the structure matrices $N_i = N_{g_i}$ for $i = 1, \dots, m$, such that:

$$B_{i\tau+1} = N_i z_\tau, i = 1,2, \dots, m. \quad (16)$$

Typically, each function g_i depends on only a subset of variables. For instance:

$$B_{\tau+1} = C_\tau \wedge D_\tau.$$

Expressing this in matrix form:

$$B_{\tau+1} = N_d C_\tau D_\tau. \quad (17)$$

To align (17) with (16), we define an auxiliary matrix:

$$E_q = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}.$$

It follows that for logical variables U, V:

$$E_q UV = V, \text{or } E_q S_{[2]} UV = U.$$

Thus, rewriting (17):

$$B_{\tau+1} = N_d E_q B_\tau C_\tau D_\tau = N_d E_q z_\tau.$$

Multiplying the equations in (16) gives:

$$z_{\tau+1} = N_1 z_\tau N_2 z_\tau \dots N_m z_\tau. \quad (18)$$

To simplify (18), we use the following:

Let $Q_k = B_1 B_2 \dots B_k$. Then:

$$Q_k^2 = \Psi_k Q_k, \quad (19)$$

where

$$\Psi_k = \prod_{j=1}^{k} I_{2^{k-j}} \otimes [(I_2 \otimes S_{[2,2^{k-j}]}) N_s \quad (20)$$

We can show this by mathematical induction. For $k = 1$:

$$Q_1^2 = B_1^2 = N_s B_1.$$

Since $S_{[2,1]} = I_2$, we have $\Psi_1 = N_s$. Assuming validity for $k = r$, we show for $k = r + 1$:

$$\begin{aligned} Q_{r+1}^2 &= B_1 B_2 \dots B_{r+1} B_1 B_2 \dots B_{r+1} \\ &= B_1 S_{[2,2^r]} B_1 [B_2 \dots B_{r+1}]^2 \\ &= (I_2 \otimes S_{[2,2^r]}) B_1^2 [B_2 \dots B_{r+1}]^2 \\ &= [(I_2 \otimes S_{[2,2^r]}) N_s] B_1 [B_2 \dots B_{r+1}]^2. \end{aligned}$$

Equation (18) can be rewritten as:

$$z_{\tau+1} = G z_\tau, \quad (21)$$

where

$$G = N_1 \prod_{j=2}^{m} [\,(I_{2^m} \otimes N_j)\Psi_m].$$

By equation (19):

$$z_\tau^2 = \Psi_m z_\tau.$$

Thus,

$$\begin{aligned} z_{\tau+1} &= N_1 z_\tau N_2 z_\tau \dots N_k z_\tau \\ &= N_1 (I_{2^n} \otimes N_2) z_\tau^2 N_3 z_\tau \cdots N_k z_\tau \\ &= N_1 (I_{2^n} \otimes N_2) \Psi_m z_\tau N_3 z_\tau \cdots N_k z_\tau \\ &= \cdots \\ &= N_1 (I_{2^n} \otimes N_2) \Psi_m (I_{2^n} \otimes N_3) \Psi_m \cdots (I_{2^n} \otimes N_k) \Psi_m z_\tau. \end{aligned}$$

2.3 Equilibrium states and cycles

We begin by examining the derivation of logical variables $\{P_i(t)\}$ from $y(t) = P_1(t)P_2(t) \dots P_n(t)$. Clearly, $y(t) \in \Omega_{2^n}$. The following has been readily established in (Cheng and Qi, 2010). Suppose $y(t) = \gamma_{2^n}^j$. Define $C_0 := 2^n - j$, then $P_k(t)$ can be computed iteratively as follows:

$$\begin{cases} P_k(t) = \lfloor \dfrac{C_{k-1}}{2^{n-k}} \rfloor \\ C_k = C_{k-1} - P_k(t) \cdot 2^{n-k}, k = 1,2, \dots, n, \end{cases} \tag{22}$$

where $\lfloor c \rfloor$ represents the greatest integer less than or equal to c. Considering the Boolean Network equation (21), let M_i, $i = 1,2, \dots, 2^n$ represent the i-th column of the network transition matrix M. It follows that$M_i \in \Omega_{2^n}$.A state $y_0 \in \Omega_{2^n}$ is termed an **equilibrium point** of system (21)

I. If $My_0 = y_0$.

II. The sequence $\{y_0, My_0, \dots, M^k y_0\}$ is a cycle of system (21) with period k, if $M^k y_0 = y_0$, and all elements in $\{y_0, My_0, \dots, M^{k-1} y_0\}$ are distinct.

Consider the Boolean network model described in equation (13). The state $\gamma_{2^m}^j$ serves as an equilibrium point if and only if, within its algebraic representation (21), the diagonal component q_{jj} of the network transition matrix Q satisfies $q_{jj} = 1$. Consequently, the total count of equilibrium states in system (13), denoted asT_c, corresponds to the number of indices j where $q_{jj} = 1$. Mathematically, this is expressed as:

$$T_c = \mathrm{Trace}(Q). \tag{23}$$

Moreover, if $q_{jj} = 1$, the corresponding column j in Q is termed a nonzero diagonal column.Furthermore, the cyclic behavior of the Boolean network system (13), was examined in (Cheng and Qi, 2010). Define a notation: For any integer $r \in \mathbb{Z}_+$, a positive integer $s \in \mathbb{Z}_+$ is a valid divisor of $r$if $s < r$and$r/s \in \mathbb{Z}_+$. The collection of all valid divisors of$r$ is denoted by$\mathcal{D}(r)$. For example, $\mathcal{D}(10) = \{1,2,5\}$ and $\mathcal{D}(15) = \{1,3,5\}$. Applying a reasoning similar to equation (23), the following result was derived.The number of cycles of length s, denoted C_s, is computed recursively as:

$$\begin{cases} C_1 = T_c \\ C_s = \dfrac{\mathrm{Trace}(Q^s) - \sum_{r\in\mathcal{D}(s)} r\, M_r}{s}, 2 \le s \le 2^m. \end{cases} \quad (24)$$

This has been proven in Cheng and Qi (2010). Regarding the upper bound for s, note that $x(t)$ can assume at most 2^m distinct states, implying that the maximum possible cycle length is at most 2^m (see Cheng and Qi (2010)). To identify these cycles, we check whether:

$$\mathrm{Trace}(Q^s) - \sum_{r\in\mathcal{D}(s)} r\, M_r > 0. \quad (25)$$

If inequality (25) holds, then s is termed a significant period.Assuming s is a significant period, let q_{jj}^s denote the (j,j)-th entry of Q^s. Define the sets:

$$E_s = \{j \mid q_{jj}^s = 1\}, s = 1,2,\dots,2^m,$$

and

$$F_s = E_s \bigcap_{r\in\mathcal{D}(s)} E_r^c,$$

where E_r^c represents the complement of E_r.Then it can be said that, let $y_0 = \gamma_{2^m}^j$. Then the sequence $\{y_0, Qy_0, \dots, Q^{s-1}y_0\}$ forms a cycle of length s if and only if$j \in F_s$.Hence, equation (24) and the above statement, outlines an efficient method for determining cycles.

2.4 Permuation-inspired method

To rigorously define this transformation, Wang et al. (2023) introduce a bijective function g that encodes Boolean network states as numerical identifiers. Let $\Omega = \{\omega_1, \dots, \omega_{2^m}\}$ denote a set containing 2^munique numerical labels. The function $g:\{0,1\}^m \to \Omega$ maps each Boolean state $(y_1, y_2, \dots, y_m)$ to a unique element in Ω. It is important to note that g is not uniquely determined; its formulation is contingent on the selection of Ω.For example:

If $\Omega = \{0,1,\dots,2^m - 1\}$, one possible bijection is the standard binary-to-decimal encoding:

$$g(y_1, y_2, \dots, y_m) = y_1 \cdot 2^{m-1} + y_2 \cdot 2^{m-2} + \cdots + y_m. \quad (26)$$

Alternatively, if $\Omega = \{1,2,\dots,2^m\}$, an alternative transformation may involve shifting the output by 1:

$$g(y_1, y_2, \dots, y_m) = y_1 \cdot 2^{m-1} + y_2 \cdot 2^{m-2} + \cdots + y_m + 1. \quad (27)$$

These examples illustrate how g can be adapted to various indexing frameworks while preserving the one-to-one correspondence between Boolean states and numerical values. The ability to redefine g highlights the connection between algebraic structures and the representation of dynamical systems.

$$g(y_1, y_2, \dots, y_m) = 2^m - (y_1(2^{m-1}) + y_2(2^{m-2}) + \cdots + y_m). \quad (28)$$

Let $h:\{0,1\}^m \to \{0,1\}^m$ be a Boolean map, and let $V_m = \{e_j\}_{j=1}^{2^m}$ be the standard basis of $\mathbb{R}^{2^m}$. Let $R = \{1,2,3,\dots,2^m\}$, and define $g:\{0,1\}^m \to R$ as a bijection map. Define the map $K:\{0,1\}^m \to V_m$ by

$$K(z) = \epsilon_{g(z)}, \mathrm{for} z \in \{0,1\}^m.$$

Then the map $\bar{h}: V_m \to V_m$ defined by

$$\bar{h} = K \circ h \circ K^{-1}$$

is isomorphic to h and can be expressed as

$$\bar{h}(v) = Mv, \text{for} v \in V_m, \quad (29)$$

where $M = [\bar{h}(e_1), \bar{h}(e_2), \dots, \bar{h}(e_{2^m})]$ (see Wang et. al (2023)).The above, indicates that instead of solving the Boolean network $z^+ = h(z)$, we can solve the linear system $w^+ = M(w)$. The map K is an arbitrary bijection from $\{0,1\}^m$ to Δ_{2^m} since g is arbitrary. Thus, the theorem suggests that any bijection from $\{0,1\}^m$ to Δ_{2^m} transforms a Boolean network system into a discrete linear dynamical system. When g is as given in Equation (28), then equation (29) shows that the linear representation obtained matches that constructed using the semi-tensor product method introduced by Cheng and Qi (2010) (see section 2.1 - 2.3). Hence, by Equation (28), the equivalent linear representation of the Boolean system $y^+ = h(y)$ is given by

$$z^+ = [e_2, e_8, e_8, e_9, e_2, e_6, e_6, e_7]z,$$

which matches the result obtained through the semi-tensor product method in section 2.1-- 2.3.Also in (Wang et. al, 2023), it was shown that, if $y_i \in \{0,1\}$ and $w_i \in \Delta_3$ be the vectors associated with y_i by equation (12) for $1 \leq i \leq m$. Then

$$w_1 \ltimes w_2 \ltimes \dots \ltimes w_m = e_{\varphi(y)} \quad (30)$$

where $\varphi(y) = 2^m - (y_1(2^{m-1}) + y_2(2^{m-2}) + \cdots + y_m) = 2^m - Dec(y)$. Let $h: \{0,1\}^m \to \{0,1\}^m$ be a Boolean map and $k: \{0,1\}^m \to P$be a bijection function as described above. We now define a new map $p: P \to P$ that is isomorphic to h as follows:

$$p = k \circ h \circ k^{-1} \quad (31)$$

The map p can be represented by the following matrix:

$$S(p) = \begin{bmatrix} p_1 & p_2 & \dots & p_{2^m} \\ p(p_1) & p(p_2) & \dots & p(p_{2^m}) \end{bmatrix} \quad (32)$$

It is important to note that $S(p)$ in Equation (32) serves as an alternate representation of the truth table. However, $S(p)$ provides a more compact form, making it easier to identify the Boolean states and their corresponding subsequent states, as well as any underlying patterns. This streamlined representation aids in the analytical study of Boolean networks, even those with an indeterminate number of nodes, this was further illustrated in (Wang et. al, 2023) with provisional examples.

2.5 Perturbed diphtheria pathogenesis network

In the study by Ugbene and Utoyo (2024), a Boolean network model was proposed to encapsulate the regulatory interactions among eight key genes implicated in diphtheria pathogenesis. This network, which captures the genetic interplay underlying the pathogenesis of diphtheria, is modeled using Boolean transition functions. To this end, boolean variables were assigned to each of the eight key genes. In the model, the variable $a_1(t)$ represents the expression of the toxin gene (Tox), $a_2(t)$ denotes the repressor gene (Rep), $a_3(t)$ corresponds to the inflammatory mediator INF1, $a_4(t)$ to INF2, $a_5(t)$ to the Toll-like receptor (TLR), $a_6(t)$ to the transcription factor AP1, $a_7(t)$ to interleukin-6 (IL6), and $a_8(t)$ to tumor necrosis factor (TNF). Here we modify, adding a gain of function to suggest the fact that TNF down-regulates TLR expression and activity, while IL-6 up-regulates TLR expression and enhances its responsiveness (Tamandl et al., 2003; Smolinska et al., 2011; Chang et al., 2014) and also define the rule that Rep is regulated by the inhibitory of Tox or the expression of either INF1 or INF2. Each gene is represented by a Boolean variablea_i (for $i = 1, \dots, 8$), which can take the value 0 (inactive) or 1 (active). The evolution of the network is governed by discrete-time updates defined by the following Boolean equations:

$$\begin{aligned}
a_1(t+1) &= \neg a_2(t),\\
a_2(t+1) &= \neg a_1(t) \vee (a_3(t) \vee a_4(t)),\\
a_3(t+1) &= a_5(t) \vee a_6(t),\\
a_4(t+1) &= a_5(t) \vee a_6(t) \vee a_7(t),\\
a_5(t+1) &= \neg a_8(t) \wedge a_7(t),\\
a_6(t+1) &= a_8(t) \vee a_7(t),\\
a_7(t+1) &= a_1(t) \vee a_3(t),\\
a_8(t+1) &= a_1(t) \vee a_6(t).
\end{aligned} \tag{33}$$

In these equations, the symbol $\neg a_i$ denotes the logical NOT of a_i,$\wedge$ denotes the logical AND, and $\vee$ denotes the logical OR.

3 Results

3.1 Conversion to a linear dynamical system via the semi-tensor product method

The semi-tensor product (STP) method is an effective algebraic technique that transforms Boolean networks into linear dynamical systems, thereby enabling the application of linear system theory to analyze network dynamics. Within this framework, each Boolean variable $x_i$is represented by a two-dimensional vector. Specifically, the Boolean value 1(true) is represented by the vector

$$\gamma_2^1 = \begin{bmatrix}1\\0\end{bmatrix},$$

and the value 0 (false) is represented by

$$\gamma_2^2 = \begin{bmatrix}0\\1\end{bmatrix}.$$

Thus, the state of each gene can be expressed in vector form.The overall state of the eight-gene network is then given by the semi-tensor (or Kronecker) product of the individual gene state vectors:

$$A = a_1 \ltimes a_2 \ltimes \cdots \ltimes a_8 \in \mathbb{R}^{2^8},$$

Where$\ltimes$denotes the semi-tensor product and A is a 256×1 vector that encapsulates the entire state of the network.Each Boolean function $f_i: \{0,1\}^8 \to \{0,1\}$ corresponding to the update rule of genex_i is then converted into an algebraic form. More precisely, there exists a structure matrix $N_i \in \mathbb{R}^{2\times 2^8}$, such that the update rule can be written as

$$a_i^+ = N_i A,$$

where the output x_i^+ is represented in vector form; that is, it is γ_2^1 if $f_i(X) = 1$ and γ_2^2 if $f_i(X) = 0$. The structure matrix M_i is constructed by first enumerating all (2^8)possible states of the network and then mapping the Boolean output of $f_i$for each state to its corresponding canonical vector.To illustrate, consider the Boolean update for gene a_1:

$$a_1^+ = \neg a_2.$$

For each of the 256 possible input states X, the value of $\neg a_2$ is computed. If the output is 1, the corresponding column of N_1 is set to γ_2^1; if the output is 0, it is set toγ_2^2. A similar procedure is applied to obtain the structure matrices $N_2, N_3, \ldots, N_8$ for the other genes.Once the structure matrices for all eight genes have been determined, the overall dynamics of the network can be encapsulated in a single linear equation. By combining the individual gene update equations, we can express the complete state transition of the network in the compact form:

$$A(t+1) = LA(t),$$

where $L \in \mathbb{R}^{2^8 \times 2^8}$ is the global transition matrix of the network. The matrix L is constructed by appropriately integrating the individual structure matrices M_I and reflects the one-step evolution of the network state.The construction of L involves defining an indexing function $h: \{0,1\}^8 \to \{1,2,\dots,256\}$ that assigns a unique integer to each Boolean state. In this way, the jth column of \(L \) corresponds to the state into which the network transitions from the jth state (as ordered by h). Although the explicit form of L is generally too cumbersome to present in full due to its size, its construction is systematic and follows directly from the STP method. Each entry in L is determined by the outputs of the structure matrices, $N_1, \dots, N_8$ corresponding to the appropriate Boolean functions evaluated at each state. Mathematically, writting the equation (33) in a structured matrix form;

$$\begin{aligned}
a_1(t+1) &= N_m a_2(t),\\
a_2(t+1) &= N_d((N_m a_1(t))(N_d a_3(t) a_4(t))),\\
a_3(t+1) &= N_d a_5(t) a_6(t),\\
a_4(t+1) &= N_d(a_5(t)(N_d a_6(t) a_7(t))),\\
a_5(t+1) &= N_c(N_m a_8(t) a_7(t)),\\
a_6(t+1) &= N_d a_8(t) a_7(t),\\
a_7(t+1) &= N_d a_1(t) a_3(t),\\
a_8(t+1) &= N_d a_1(t) a_6(t).
\end{aligned} \tag{34}$$

So that,

$$\begin{aligned}
a_1(t+1) &= N_m a_2(t),\\
a_2(t+1) &= N_d(N_m(I_2 \otimes N_d) a_1(t) a_3(t) a_4(t)),\\
a_3(t+1) &= N_d a_5(t) a_6(t),\\
a_4(t+1) &= N_d(I_2 \otimes N_d) a_5(t) a_6(t) a_7(t),\\
a_5(t+1) &= N_c(N_m a_8(t) a_7(t)),\\
a_6(t+1) &= N_d a_8(t) a_7(t),\\
a_7(t+1) &= N_d a_1(t) a_3(t),\\
a_8(t+1) &= N_d a_1(t) a_6(t).
\end{aligned} \tag{35}$$

Setting $A(t) = a_1(t)a_2(t)a_3(t)a_4(t)a_5(t)a_6(t)a_7(t)a_8(t)$, we can calculate L as

$$\begin{aligned}
&A(t+1) = N_m(I_2 \otimes N_d(N_m(I_2 \otimes N_d)))(I_{2^4} \otimes N_d)(I_{2^6} \otimes N_d(I_2 \otimes N_d))(I_{2^7} \otimes N_c N_m)\\
&(I_{2^8} \otimes (I_{2^8} \otimes (I_{2^8} \otimes N_d)))S_{[2]}(I_{2^5} \otimes S_{[2]})(I_{2^4} \otimes N_s)(I_{2^5} \otimes N_s)(I_{2^7} \otimes S_{[2]})\\
&(I_{2^7} \otimes N_s)(I_{2^8} \otimes N_s)(I_{2^7} \otimes S_{[2]})(I_{2^7} \otimes N_s)(I_{2^8} \otimes N_s)(I_{2^8} \otimes S_{[2]})\\
&(I_{2^7} \otimes S_{[2,2^7]})(I_2 \otimes N_s)(I_{2^5} \otimes S_{[2,2^5]})(I_{2^2} \otimes N_s)(I_{2^2} \otimes S_{[2,2^2]})\\
&(I_{2^5} \otimes N_s) a_1(t)a_2(t)a_3(t)a_4(t)a_5(t)a_6(t)a_7(t)a_8(t)
\end{aligned} \tag{36}$$

Hence

$$\begin{aligned}
&L = N_m(I_2 \otimes N_d(N_m(I_2 \otimes N_d)))(I_{2^4} \otimes N_d)(I_{2^6} \otimes N_d(I_2 \otimes N_d))(I_{2^7} \otimes N_c N_m)\\
&(I_{2^8} \otimes (I_{2^8} \otimes (I_{2^8} \otimes N_d)))S_{[2]}(I_{2^5} \otimes S_{[2]})(I_{2^4} \otimes N_s)(I_{2^5} \otimes N_s)(I_{2^7} \otimes S_{[2]})\\
&(I_{2^7} \otimes N_s)(I_{2^8} \otimes N_s)(I_{2^7} \otimes S_{[2]})(I_{2^7} \otimes N_s)(I_{2^8} \otimes N_s)(I_{2^8} \otimes S_{[2]})(I_{2^7} \otimes S_{[2,2^7]})\\
&(I_2 \otimes N_s)(I_{2^5} \otimes S_{[2,2^5]})(I_{2^2} \otimes N_s)(I_{2^2} \otimes S_{[2,2^2]})(I_{2^5} \otimes N_s)
\end{aligned} \tag{37}$$

Then this can be solved as

$$L = \gamma_{256}[41943193681966719541963195681966719516206152058020879207162081520780208792071620815207802087920716208152078020879207144206143205208208207207144208143207208208207207144206143205208208207207144208143207208208207207602505924912425212325160252592511242521232564254632531282561272556425663255128256127255192254191253256256255255192256191255256256255255192254191253256256255255192256191255256256255255362263522510022899227362283522710022899227482384723711224011123948240472391122401112748238472371122401112394824047239112240111239176236175237240240239239176240175239240240239239176236175237240240239239176240175239240240239239442344323310823610723544236432351082361072354423643235108236107235482384723711224011123948240472391122401112391762381752372402402392391762401752392402402391762381752372402402392391762401752392402402392391762381752372402402392391762401752392402402392391]$$

When we mapped out the system's behavior using semi-tensor product (STP) analysis, what stood out was how predictable it all became. No chaos, no endless loops, just two clear endpoints it always settles into. The first,

γ_{256}^{239} (corresponding to state 01110111) with a large basin of 254 states,

acts like a giant magnet. Almost every starting point, 254 out of 256!, gets pulled here. Imagine a bustling city: immune sensors like TLR are on high alert, cytokines like IL6 and TNF are shouting orders, and interferon signals (INF1/INF2) are flashing warnings. But here's the kicker, the toxin gene stays silent. It's like the body's defense team is working overtime, but without friendly fire. Then there's

γ_3^3 (corresponding to state 01000000) with a small basin of 2 states,

the quiet cousin. Only two paths lead here, and it's a ghost town in comparison. Just INF1 flickering faintly, like a lone nightlight, while everything else, TLR, AP1, IL6, TNF, shuts down. This attractor feels fragile, like balancing a pencil on its tip. Breathe too hard, and it topples back into Attractor 1. What's cool, and a bit surprising, is that the system doesn't cycle. No back-and-forth, no pendulum swings. It's all-or-nothing: either full-throttle immune mode or near-silence. This "bistability" tells us the system's wired for extremes, not middle grounds.

Using STP here wasn't just math gymnastics. It's like reverse-engineering the system's rulebook. We saw why Attractor 1 dominates (it's got all the reinforcement loops, think IL6 propping up TLR, which feeds back into IL6) and why Attractor 2 is a fluke (no backup, no safety nets). For folks designing treatments, this is huge. Want to keep the body in defense mode? Boost those feedback loops. Need to calm an overzealous response? Maybe tweak INF1 to nudge things toward quiet, but good luck, since it's like herding cats. Bottom line? Tools like STP turn abstract networks into something we can actually work with. They're not just for theorists, they help us see where to push, pull, or patch things up. And in a world where pathogens play dirty, that's a pretty solid advantage.

3.2 Conversion via the permutation method

Here, we use the permutation-based technique to examine Boolean maps over finite state spaces, offering a clear window into the system's dynamics. We start with a Boolean function

$$p: \{0,1\}^8 \to \{0,1\}^8,$$

and we work with the canonical basis of $\mathbb{R}^{2^8}$, denoted by

$$V_8 = \{e_1, e_2, \dots, e_{2^8}\}.$$

To uniquely identify each 8-bit vector, we introduce a bijection

$$g: \{0,1\}^8 \to \{1,2, \dots, 2^8\},$$

which assigns a distinct index to every element of $\{0,1\}^8$. Using this bijection, we define a mapping

$$K(z) = \epsilon_{g(z)} \text{ for} z \in \{0,1\}^8,$$

where $\epsilon_{g(z)}$ denotes the corresponding standard basis vector in V_8.

Table 1 An isomorphic mapping of the state transitions in the Boolean network (All Possible State Transitions).

p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$
e_1	e_4	e_2	e_{194}	e_3	e_3	e_4	e_{193}	e_5	e_{68}	e_6	e_{196}	e_7	e_{67}	e_8	e_{195}
e_9	e_4	e_{10}	e_{196}	e_{11}	e_3	e_{12}	e_{195}	e_{13}	e_{68}	e_{14}	e_{196}	e_{15}	e_{67}	e_{16}	e_{195}
e_{17}	e_{16}	e_{18}	e_{206}	e_{19}	e_{15}	e_{20}	e_{205}	e_{21}	e_{80}	e_{22}	e_{208}	e_{23}	e_{79}	e_{24}	e_{207}
e_{25}	e_{16}	e_{26}	e_{208}	e_{27}	e_{15}	e_{28}	e_{207}	e_{29}	e_{80}	e_{30}	e_{208}	e_{31}	e_{79}	e_{32}	e_{207}
e_{33}	e_{144}	e_{34}	e_{206}	e_{35}	e_{143}	e_{36}	e_{205}	e_{37}	e_{208}	e_{38}	e_{208}	e_{39}	e_{207}	e_{40}	e_{207}
e_{41}	e_{144}	e_{42}	e_{208}	e_{43}	e_{143}	e_{44}	e_{207}	e_{45}	e_{208}	e_{46}	e_{208}	e_{47}	e_{207}	e_{48}	e_{207}
e_{49}	e_{144}	e_{50}	e_{206}	e_{51}	e_{143}	e_{52}	e_{205}	e_{53}	e_{208}	e_{54}	e_{208}	e_{55}	e_{207}	e_{56}	e_{207}
e_{57}	e_{144}	e_{58}	e_{208}	e_{59}	e_{143}	e_{60}	e_{207}	e_{61}	e_{208}	e_{62}	e_{208}	e_{63}	e_{207}	e_{64}	e_{207}
e_{65}	e_{60}	e_{66}	e_{250}	e_{67}	e_{59}	e_{68}	e_{249}	e_{69}	e_{124}	e_{70}	e_{252}	e_{71}	e_{123}	e_{72}	e_{251}
e_{73}	e_{60}	e_{74}	e_{252}	e_{75}	e_{59}	e_{76}	e_{251}	e_{77}	e_{124}	e_{78}	e_{252}	e_{79}	e_{123}	e_{80}	e_{251}
e_{81}	e_{64}	e_{82}	e_{254}	e_{83}	e_{63}	e_{84}	e_{253}	e_{85}	e_{128}	e_{86}	e_{256}	e_{87}	e_{127}	e_{88}	e_{255}
e_{89}	e_{64}	e_{90}	e_{256}	e_{91}	e_{63}	e_{92}	e_{255}	e_{93}	e_{128}	e_{94}	e_{256}	e_{95}	e_{127}	e_{96}	e_{255}
e_{97}	e_{192}	e_{98}	e_{254}	e_{99}	e_{191}	e_{100}	e_{253}	e_{101}	e_{256}	e_{102}	e_{256}	e_{103}	e_{255}	e_{104}	e_{255}
e_{105}	e_{192}	e_{106}	e_{256}	e_{107}	e_{191}	e_{108}	e_{255}	e_{109}	e_{256}	e_{110}	e_{256}	e_{111}	e_{255}	e_{112}	e_{255}
e_{113}	e_{192}	e_{114}	e_{254}	e_{115}	e_{191}	e_{116}	e_{253}	e_{117}	e_{256}	e_{118}	e_{256}	e_{119}	e_{255}	e_{120}	e_{255}

e_{121}	e_{192}	e_{122}	e_{256}	e_{123}	e_{191}	e_{124}	e_{255}	e_{125}	e_{256}	e_{126}	e_{256}	e_{127}	e_{255}	e_{128}	e_{255}
e_{129}	e_{36}	e_{130}	e_{226}	e_{131}	e_{35}	e_{132}	e_{225}	e_{133}	e_{100}	e_{134}	e_{228}	e_{135}	e_{99}	e_{136}	e_{227}
e_{137}	e_{36}	e_{138}	e_{228}	e_{139}	e_{35}	e_{140}	e_{227}	e_{141}	e_{100}	e_{142}	e_{228}	e_{143}	e_{99}	e_{144}	e_{227}
e_{145}	e_{48}	e_{146}	e_{238}	e_{147}	e_{47}	e_{148}	e_{237}	e_{149}	e_{112}	e_{150}	e_{240}	e_{151}	e_{111}	e_{152}	e_{239}
e_{153}	e_{48}	e_{154}	e_{240}	e_{155}	e_{47}	e_{156}	e_{239}	e_{157}	e_{112}	e_{158}	e_{240}	e_{159}	e_{111}	e_{160}	e_{239}
e_{161}	e_{176}	e_{162}	e_{236}	e_{163}	e_{175}	e_{164}	e_{237}	e_{165}	e_{240}	e_{166}	e_{240}	e_{167}	e_{239}	e_{168}	e_{239}
e_{169}	e_{176}	e_{170}	e_{240}	e_{171}	e_{175}	e_{172}	e_{239}	e_{173}	e_{240}	e_{174}	e_{240}	e_{175}	e_{239}	e_{176}	e_{239}
e_{177}	e_{176}	e_{178}	e_{236}	e_{179}	e_{175}	e_{180}	e_{237}	e_{181}	e_{240}	e_{182}	e_{240}	e_{183}	e_{239}	e_{184}	e_{239}
e_{185}	e_{176}	e_{186}	e_{240}	e_{187}	e_{175}	e_{188}	e_{239}	e_{189}	e_{240}	e_{190}	e_{240}	e_{191}	e_{239}	e_{192}	e_{239}
e_{193}	e_{44}	e_{194}	e_{234}	e_{195}	e_{43}	e_{196}	e_{233}	e_{197}	e_{108}	e_{198}	e_{236}	e_{199}	e_{107}	e_{200}	e_{235}
e_{201}	e_{44}	e_{202}	e_{236}	e_{203}	e_{43}	e_{204}	e_{235}	e_{205}	e_{108}	e_{206}	e_{236}	e_{207}	e_{107}	e_{208}	e_{235}
e_{209}	e_{48}	e_{210}	e_{238}	e_{211}	e_{47}	e_{212}	e_{237}	e_{213}	e_{112}	e_{214}	e_{240}	e_{215}	e_{111}	e_{216}	e_{239}
e_{217}	e_{48}	e_{218}	e_{240}	e_{219}	e_{47}	e_{220}	e_{239}	e_{221}	e_{112}	e_{222}	e_{240}	e_{223}	e_{111}	e_{224}	e_{239}
e_{225}	e_{176}	e_{226}	e_{238}	e_{227}	e_{175}	e_{228}	e_{237}	e_{229}	e_{240}	e_{230}	e_{240}	e_{231}	e_{239}	e_{232}	e_{239}
e_{233}	e_{176}	e_{234}	e_{240}	e_{235}	e_{175}	e_{236}	e_{239}	e_{237}	e_{240}	e_{238}	e_{240}	e_{239}	e_{239}	e_{240}	e_{239}
e_{241}	e_{176}	e_{242}	e_{238}	e_{243}	e_{175}	e_{244}	e_{237}	e_{245}	e_{240}	e_{246}	e_{240}	e_{247}	e_{239}	e_{248}	e_{239}
e_{249}	e_{176}	e_{250}	e_{240}	e_{251}	e_{175}	e_{252}	e_{239}	e_{253}	e_{240}	e_{254}	e_{240}	e_{255}	e_{239}	e_{256}	e_{239}

Table 2 Essential state transitions.

p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$
e_3	e_3	e_4	e_{193}	e_{15}	e_{67}	e_{16}	e_{195}	e_{35}	e_{143}	e_{36}	e_{205}	e_{43}	e_{143}	e_{44}	e_{207}

e_{47}	e_{207}	e_{48}	e_{207}	e_{59}	e_{143}	e_{50}	e_{207}	e_{63}	e_{207}	e_{64}	e_{207}	e_{67}	e_{59}	e_{68}	e_{249}
e_{79}	e_{123}	e_{80}	e_{251}	e_{99}	e_{191}	e_{100}	e_{253}	e_{107}	e_{191}	e_{108}	e_{255}	e_{111}	e_{255}	e_{112}	e_{255}
e_{123}	e_{191}	e_{124}	e_{255}	e_{127}	e_{255}	e_{128}	e_{255}	e_{143}	e_{99}	e_{144}	e_{227}	e_{175}	e_{239}	e_{176}	e_{239}
e_{191}	e_{239}	e_{192}	e_{239}	e_{193}	e_{44}	e_{194}	e_{234}	e_{195}	e_{43}	e_{196}	e_{233}	e_{205}	e_{108}	e_{206}	e_{236}
e_{207}	e_{107}	e_{208}	e_{235}	e_{225}	e_{176}	e_{226}	e_{238}	e_{227}	e_{175}	e_{228}	e_{237}	e_{233}	e_{176}	e_{234}	e_{240}
e_{235}	e_{175}	e_{236}	e_{239}	e_{237}	e_{240}	e_{238}	e_{240}	e_{239}	e_{239}	e_{240}	e_{239}	e_{249}	e_{176}	e_{250}	e_{240}
e_{251}	e_{175}	e_{252}	e_{239}	e_{253}	e_{240}	e_{254}	e_{240}	e_{255}	e_{239}	e_{256}	e_{239}				

Table 3 Paths to stability.

p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$
e_3	e_3	e_{43}	e_{143}	e_{44}	e_{207}	e_{59}	e_{143}	e_{67}	e_{59}	e_{99}	e_{191}	e_{107}	e_{191}	e_{108}	e_{255}
e_{123}	e_{191}	e_{143}	e_{99}	e_{175}	e_{239}	e_{176}	e_{239}	e_{191}	e_{239}	e_{193}	e_{44}	e_{195}	e_{43}	e_{205}	e_{108}
e_{207}	e_{107}	e_{227}	e_{175}	e_{233}	e_{176}	e_{234}	e_{240}	e_{235}	e_{175}	e_{236}	e_{239}	e_{237}	e_{240}	e_{238}	e_{240}
e_{239}	e_{239}	e_{240}	e_{239}	e_{249}	e_{176}	e_{251}	e_{175}	e_{253}	e_{240}	e_{255}	e_{239}				

Table 4 Roadmap to attractors.

p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$
e_3	e_3	e_{43}	e_{143}	e_{44}	e_{207}	e_{59}	e_{143}	e_{99}	e_{191}	e_{107}	e_{191}	e_{108}	e_{255}	e_{143}	e_{99}
e_{175}	e_{239}	e_{176}	e_{239}	e_{191}	e_{239}	e_{207}	e_{107}	e_{239}	e_{239}	e_{240}	e_{239}	e_{255}	e_{239}		

Table 5 Limit sets.

p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$	p	$S(p)$
e_3	e_3	e_{99}	e_{191}	e_{107}	e_{191}	e_{143}	e_{99}	e_{191}	e_{239}	e_{207}	e_{107}	e_{239}	e_{239}	e_{255}	e_{239}

Next, we transform the original function p by considering

$$S(p) = K \circ p \circ K^{-1}.$$

This re-expression of p in the space V_8 preserves its dynamic behavior, but now in a linear algebraic form. In fact, we can write

$$S(p) = Mp, \qquad \text{for} v \in V_8,$$

where the matrix$M$ is constructed as

$$M = [S(e_1), S(e_2), \dots, S(e_{2^8})]. \tag{38}$$

As noted in Wang et al. (2023), this approach offers a straightforward linear representation of Boolean dynamics, making it easier to analyze state transitions. A particularly useful aspect of this method is that it enables us to remove nonessential states specifically, the transient states and those with no predecessors (often referred to as Garden-of-Eden states). For example, while our complete state transition (see Table 1) includes every state, Table 2, presents the system after these Garden-of-Eden states have been eliminated. Moreover, Tables 3 and 4 detail both the transient states and those that eventually contribute to the system's long-term behavior, with Table 5, isolating the final limit set.By filtering out these extraneous states, we gain a much clearer understanding of how the system ultimately settles into its attractors and the stability of its dynamics. This refined perspective not only streamlines our analysis but also enhances our insight into the structural properties of Boolean networks. The implications of these findings extend to various fields where understanding complex, discrete dynamic systems is essential.

4 Discussion

The findings from previous section has revealed profound insights into the regulatory logic of a Boolean network modeling diphtheria pathogenesis, where genes are encoded in the order Tox (toxin production), Rep (toxin repressor), INF1/INF2 (interferon signaling), TLR (Toll-like receptor), AP1 (transcription factor), IL6 (interleukin-6), and TNF (tumor necrosis factor). By redefining the regulatory rules governing Rep, such that its activation is controlled either by the inhibition of Tox or the expression of INF1/INF2, the system was shown to converge exclusively to pathogen-free states, where toxin production is permanently silenced. This critical modification not only eliminated cycles in the network but also ensured that all transient states eventually funneled into stable attractors devoid of pathogenic activity. The absence of cycles underscores the system’s evolutionary design to prioritize stability and avoid oscillatory behaviors, which could otherwise lead to unpredictable or harmful immune responses. Biologically, this reflects a fail-safe mechanism where the host’s immune system is wired to suppress toxin production aggressively. The dual regulation of Rep, triggered either by direct inhibition of Tox or by interferon signaling (INF1/INF2), mimics real-world immune strategies where redundancy ensures robustness. For instance, even if Tox evades direct repression (e.g., through mutations), interferon signaling acts as a backup to activate Rep, ensuring toxin suppression. This redundancy aligns with the evolutionary arms race between pathogens and hosts, where layered defenses are critical for survival. The convergence to non-pathogenic states suggests that the network is intrinsically biased toward host protection, prioritizing toxin silencing over other outcomes. Furthermore, we had explored the role of TLR, a key immune sensor, by analyzing its overexpression and suppression. Overexpression of TLR, simulating hyperactive immune sensing—resulted in a single attractor (01111111) with a basin of 128 states, exactly half the size of the original basin observed under normal conditions. This attractor represents a hyper-inflammatory state where TLR, INF1/INF2, IL6, and TNF are fully active, while Rep remains inactive (Tox = 0). Paradoxically, despite Rep being off, toxin production is absent here, likely due to the

overwhelming immune activity indirectly suppressing Tox through other pathways. The halved basin size indicates reduced robustness, implying that hyperactive immunity destabilizes the system, making it more sensitive to perturbations. Conversely, suppressing TLR yielded the same attractors as the modified rules in earlier analyses, but with basins of identical size to the overexpression scenario. This symmetry suggests that TLR activity, whether excessive or absent, constrains the system's flexibility, forcing it into a narrower range of stable states.

These results have critical implications for understanding immune-pathogen interactions. The network's preference for non-pathogenic states, even under extreme conditions like TLR overexpression, highlights the prioritization of host survival. The system's architecture ensures that toxin production is suppressed unless multiple safeguards fail simultaneously. For example, the requirement for both INF1/INF2 and Rep inactivation to activate Tox creates a high threshold for pathogenicity, reducing the likelihood of accidental or transient toxin release. This aligns with biological systems where critical virulence factors are tightly regulated to avoid unnecessary host damage, which could otherwise compromise transmission or invite immune retaliation. The absence of cycles in the modified network further emphasizes its stability. In biological terms, cycles could represent harmful oscillations, such as recurrent inflammation or periodic toxin production, which would destabilize the host-pathogen equilibrium. The elimination of such behavior suggests that the network is optimized for monotonic convergence, once the system commits to a trajectory (e.g., immune activation or toxin suppression), it does not reverse course. This is analogous to "point-of-no-return" mechanisms in apoptosis or cell differentiation, where transitions are irreversible to ensure decisive outcomes. Practically, these findings offer a blueprint for therapeutic interventions. For instance, drugs that mimic INF1/INF2 signaling could reinforce Rep activation, ensuring toxin suppression even in strains where Tox mutations evade direct repression. Similarly, modulating TLR activity, either enhancing or suppressing it, could help calibrate immune responses. The reduced basin size under TLR overexpression warns against therapies that hyperstimulate immune sensors, as this could shrink the system's resilience to perturbations. Conversely, TLR suppression might be useful in curbing excessive inflammation without compromising toxin control, given that the core attractors remain unchanged. This also raises questions about evolutionary trade-offs. The network's robustness against toxin production likely comes at a cost, such as energetic demands from sustained immune activity or vulnerability to pathogens that exploit interferon pathways. Future work could explore how these trade-offs shape pathogen evolution, for example, whether diphtheria evolves mutations to disrupt INF1/INF2-mediated Rep activation or to decouple TLR activity from downstream cytokine production. In broader terms, this work exemplifies how Boolean modeling can unravel the logic of biological networks. By simplifying complex interactions into binary states, the model distills essential regulatory principles, redundancy, stability, and hierarchical control, that govern real-world systems. These principles are not unique to diphtheria; they resonate across immune networks, synthetic biology, and even ecological systems where stability and redundancy determine resilience. The methodology, particularly the permutation-based elimination of non-essential states, could be adapted to study other pathogens or cellular processes, offering a generalizable scheme for dissecting complexity.

Ultimately, the study underscores a fundamental truth: biological systems are not just collections of components but orchestrated networks where structure dictates function. The precise arrangement of feedback loops, redundancy, and fail-safes in this diphtheria model reveals how evolution sculpts networks to balance aggression and restraint, ensuring survival in a chaotic world. For researchers, this means that tweaking individual nodes (e.g., Rep or TLR) can have ripple effects across the entire system, demanding holistic approaches to intervention. For clinicians, it reinforces the importance of understanding pathogen networks as dynamic systems, not static targets, a perspective that could transform how we design treatments for infectious

diseases.

5 Conclusion

The findings from this study provide a robust foundation for understanding and controlling the diphtheria pathogenesis network. By demonstrating that redefining the regulatory logic of the Rep gene, linking its activation to either the inhibition of Tox or the expression of INF1/INF2, ensures convergence to pathogen-free states, we highlight a critical mechanism for silencing toxin production. This regulatory redundancy, combined with the system's cycle-free architecture, underscores its evolutionary optimization for stability and host protection. Furthermore, perturbations such as TLR overexpression or suppression reveal how immune hyperactivity or passivity constrains the system's flexibility, narrowing its attractor landscape. These insights lay the groundwork for therapeutic strategies that exploit the network's inherent logic to enforce toxin suppression or modulate immune responses.To implement these insights, future studies can leverage linear algebraic representations of Boolean dynamics for targeted control design. A Boolean control network can be conceptualized as a set of interconnected logical rules governing gene states, where inputs (e.g., drugs, cytokines) modulate transitions between states. For instance, consider a system with genes $A_1, A_2, \dots, A_n$, control inputs $u_1, u_2, \dots, u_m$ (e.g., therapeutic agents), and outputs $y_1, y_2, \dots, y_p$ (e.g., biomarkers). The system's evolution can be described as:

$$\begin{cases} A_i(t+1) = f_i(A_1(t), \dots, A_n(t), u_1(t), \dots, u_m(t)), i = 1, \dots, n \\ y_j(t) = h_j(A_1(t), \dots, A_n(t)), j = 1, \dots, p \end{cases} \tag{39}$$

Here, f_i and h_j are logical functions encoding regulatory interactions. By combining state variables, controls, and outputs into composite vectors, $x(t)$(states), $u(t)$ (controls), and $y(t)$ (outputs), the system can be translated into a linear algebraic form:

$$\begin{cases} x(t+1) = L \cdot u(t) \ltimes x(t) \\ y(t) = H \cdot x(t) \end{cases} \tag{40}$$

where L and H are structured matrices capturing the system's logic. Crucially, this linearization enables the use of computational tools like the semi-tensor product (STP) to calculate control-dependent transitions and design interventions. Unlike brute-force methods, STP efficiently computes the time-varying transition matrix $L(t) = L_u \ltimes x(t)$, which maps current states to future states under specific controls. This approach not only identifies viable control sequences but also clarifies input-output relationships, making it possible to pinpoint therapeutic targets that steer the network toward desired outcomes (e.g., toxin suppression). By bridging Boolean modeling with control theory, this approach transforms abstract network dynamics into actionable strategies. It offers a blueprint for precision medicine in infectious diseases, where interventions are designed not just to inhibit pathogens but to reprogram host-pathogen interactions at a systems level. Future work could extend this methodology to other virulence networks, opening avenues for universally applicable control paradigms in infectious disease management. This method reframes therapeutic design as a network control problem, where diseases are treated by steering biochemical systems toward health-associated attractors. Unlike conventional "one-drug-one-target" approaches, it embraces the complexity of biological networks, offering a path to therapies that are as adaptive and layered as the systems they aim to control. For diphtheria, this could mean combinatorial regimens that silence toxins, recalibrate immunity, and preempt resistance, all by leveraging the system's own logic against it. In essence, we move from fighting pathogens to orchestrating a biochemical symphony where host protection is the final, inevitable note.

References

Belsey MA, Sinclair M, Roder MR, LeBlanc DR. 1969. Corynebacterium diphtheriae skin infections in alabama and louisiana: A factor in the epidemiology of diphtheria. The New England Journal of Medicine, 280(3): 135-141

Chang YL, Chen TH, Wu YH, Chen GA, et al. 2014. A novel tlr2-triggered signalling crosstalk synergistically intensifies tnf-mediated il-6 induction. Journal of Cellular and Molecular Medicine, 18: 1344-1357

Cheng D, Qi H. 2005. Matrix expression of logic and fuzzy control. In: Proceedings of the 44th IEEE Conference on Decision and Control. 3273-3278

Cheng D, Qi H. 2010. A linear representation of dynamics of boolean networks. IEEE Transactions on Automatic Control, 55(10): 2251-2258

Farrow C, Heidel J, Maloney H, Rogers J. 2004. Scalar equations for synchronous boolean networks with biological applications. IEEE Transactions on Neural Networks, 15(2): 348-354

Hadfield TL, McEvoy P, Polotsky Y, Tzinserling VA, Yakovlev AA. 2000. The pathology of diphtheria. The Journal of Infectious Diseases, 181(Suppl 1): S116-S120

Harris SE, Sawhill BK, Wuensche A, Kauffman S. 2002. A model of transcriptional regulatory networks based on biases in the observed regulation rules. Complexity, 7: 23-40

Holmes RK. 2000. Biology and molecular epidemiology of diphtheria toxin and the tox gene. The Journal of Infectious Diseases, 181(Suppl 1): S156-S167

Kashiwagi Y, Miyata A, Kumagai T, Maehara K, et al. 2014. Production of inflammatory cytokines in response to diphtheria-pertussis-tetanus (dpt), haemophilus influenzae type b (hib), and 7-valent pneumococcal (pcv7) vaccines. Human Vaccines and Immunotherapeutics, 10(3): 677-685

Kayoh C, Ugbene I. 2023. Towards mechanism-based interventions for auxin signaling: application of algebraic edge control to a boolean network model. International Journal of Mathematical Analysis and Modelling, 6(2): 1-10

Kolibo DV, Romaniuk SI. 2001. Matematicheskaia model' infektsionnogo protsessa pri difterii dlia opredeleniia terapevticheskoĭ dozy antitoksicheskoĭ protivodifteriĭnoĭ syvorotki. Ukrains'kyi Biokhimichnyi Zhurnal, 73(2): 144-151

Menini L, Possieri C, Tornambè A. 2019. Boolean network analysis through the joint use of linear algebra and algebraic geometry. Journal of Theoretical Biology, 472: 46-53

Moehring T, Moehring J, Stinebring W. 1971. Response of interferon-treated cells to diphtheria toxin. Infection and Immunity, 4: 747-752

Pimenta FP, Matias GA, Pereira GA, Camello TC, et al. 2008a. A pcr for dtxr gene: Application to diagnosis of non-toxigenic and toxigenic corynebacterium diphtheriae. Molecular and Cellular Probes, 22(3): 189-192

Pimenta FP, Souza MC, Pereira GA, et al. 2008b. Dnase test as a novel approach for the routine screening of corynebacterium diphtheriae. Letters in Applied Microbiology, 46(3): 307-311

Possieri C, Teel AR. 2017. Asymptotic stability in probability for stochastic boolean networks. Automatica, 83: 1-9

Qi H, Cheng D. 2008. Logic and logic-based control. Journal of Control Theory and Applications, 6(1): 123-133

Qu H, Yuan Q, Pang J, Mizera A. 2015. Improving bdd-based attractor detection for synchronous boolean networks. In: Proceedings of the 7th Asia-Pacific Symposium on Internetware, Internetware '15. 212-220, Association for Computing Machinery

Rade L, Westergren B. 1998. Mathematics Handbook for Science and Engineering (4th edition). Studentlitteratur, Sweden

Schmitt M, Twiddy E, Holmes R. 1992. Purification and characterization of the diphtheria toxin repressor. Proceedings of the National Academy of Sciences of the United States of America, 89(16): 7576-7580

Shu H, Zhou J, Lian Q, Li H, Zhao D, Zeng J, Ma J. 2021. Modeling gene regulatory networks using neural network architectures. Nature Computational Science, 1: 491-501

Smolinska M, Page T, Urbaniak A, Mutch BE, Horwood N. 2011. Hck tyrosine kinase regulates tlr4-induced tnf and il-6 production via ap-1. The Journal of Immunology, 187: 6043-6051

Tamandl D, Bahrami M, Wessner B, Weigel G, et al. 2003. Modulation of toll-like receptor 4 expression on human monocytes by tumor necrosis factor and interleukin-6: Tumor necrosis factor evokes lipopolysaccharide hyporesponsiveness, whereas interleukin-6 enhances lipopolysaccharide activity. Shock, 20: 224-229

Ugbene IJ, Agwemuria RO. 2024. Identifying control targets for regulating mild cognitive impairment using reduced computational models of a life kinetic network. FUPRE Journal of Scientific and Industrial Research, 1(1): 1-10

Ugbene IJ, Utoyo OT. 2024. Determining synchronously updated fixed points and attractors in a prototype boolean gene regulation model of diphtheria pathogenesis. Open Journal of Mathematical Science, 8: 167-184

Ugbene JI, Utoyo-Ovokaeefe T. 2024. Switching stomatal aperture dynamics through computationally algebraic node control. FUPRE Journal of Scientific and Industrial Research, 1(2): 1-10

Veliz-Cuba A, Aguilar B, Hinkelmann F, Laubenbacher R. 2014. Steady state analysis of boolean molecular network models via model reduction and computational algebra. BMC Bioinformatics, 15

Wang B, Li H, Yu Y. 2020. On detectability of boolean control networks. Nonlinear Analysis: Hybrid Systems, 36: 100859

Wang RS, Albert R. 2009. Discrete dynamical modeling of cellular signaling networks. Methods in Enzymology, 467: 281-306

Wang Y, Leite MCA, Ben-Tal A. 2023. From boolean networks to linear dynamical systems: a simplified route. Journal of Difference Equations and Applications, 29(5): 542-560

Wilson SP, James SS, Whiteley DJ, Krubitzer LA. 2019. Limit cycle dynamics can guide the evolution of gene regulatory networks towards point attractors. Scientific Reports, 9(1): 16750

Yang J, Lu J, Lou J, Liu Y. 2020. Synchronization of drive-response boolean control networks with impulsive disturbances. Applied Mathematics and Computation, 364: 124679

Yu Y, Feng J, Pan J, Cheng D. 2019. Block decoupling of boolean control networks. IEEE Transactions on Automatic Control, 64(8): 3129-3140

Yuan Q, Mizera A, Pang J, Qu H. 2019. A new decomposition-based method for detecting attractors in synchronous boolean networks. Scientific Computing and Programming, 180: 18-35

Zhang WJ. 2018. Fundamentals of Network Biology. World Scientific Europe, London, UK

Appendix

Table 6 boolean rules for the network.

Node	Boolean Transition Rules
Tox	NOT Rep
Rep	NOT Tox OR (INF1 OR INF2)
INF1	TLR OR AP1
INF2	TLR OR AP1 OR IL6
TLR	NOT TNF AND IL6
AP1	TNF OR IL6
IL6	Tox OR INF1
TNF	Tox OR AP1

Table 7 Attractors and their basins.

State	Next State	Attr. basin	#trans. to attr
$\epsilon_1 = 00000000$	$\epsilon_4 = 11000000$	1	7
$\epsilon_2 = 10000000$	$\epsilon_{194} = 10000011$	1	4
$\epsilon_3 = 01000000$	$\epsilon_3 = 01000000$	2	0
$\epsilon_4 = 11000000$	$\epsilon_{193} = 00000011$	1	6
$\epsilon_5 = 00100000$	$\epsilon_{68} = 11000010$	1	4
$\epsilon_6 = 10100000$	$\epsilon_{196} = 11000011$	1	4
$\epsilon_7 = 01100000$	$\epsilon_{67} = 01000010$	1	6
$\epsilon_8 = 11100000$	$\epsilon_{195} = 01000011$	1	6
$\epsilon_9 = 00010000$	$\epsilon_4 = 11000000$	1	7
$\epsilon_{10} = 10010000$	$\epsilon_{196} = 11000011$	1	4
$\epsilon_{11} = 01010000$	$\epsilon_3 = 01000000$	2	1
$\epsilon_{12} = 11010000$	$\epsilon_{195} = 01000011$	1	6
$\epsilon_{13} = 00110000$	$\epsilon_{68} = 11000010$	1	4
$\epsilon_{14} = 10110000$	$\epsilon_{196} = 11000011$	1	4
$\epsilon_{15} = 01110000$	$\epsilon_{67} = 01000010$	1	6
$\epsilon_{16} = 11110000$	$\epsilon_{195} = 01000011$	1	6
$\epsilon_{17} = 00001000$	$\epsilon_{16} = 11110000$	1	7
$\epsilon_{18} = 10001000$	$\epsilon_{206} = 10110011$	1	3
$\epsilon_{19} = 01001000$	$\epsilon_{15} = 01110000$	1	7
$\epsilon_{20} = 11001000$	$\epsilon_{205} = 00110011$	1	4
$\epsilon_{21} = 00101000$	$\epsilon_{80} = 11110010$	1	4
$\epsilon_{22} = 10101000$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{23} = 01101000$	$\epsilon_{79} = 01110010$	1	4
$\epsilon_{24} = 11101000$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{25} = 00011000$	$\epsilon_{16} = 11110000$	1	7
$\epsilon_{26} = 10011000$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{27} = 01011000$	$\epsilon_{15} = 01110000$	1	7
$\epsilon_{28} = 11011000$	$\epsilon_{207} = 01110011$	1	4

$\epsilon_{29} = 00111000$	$\epsilon_{80} = 11110010$	1	4
$\epsilon_{30} = 10111000$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{31} = 01111000$	$\epsilon_{79} = 01110010$	1	4
$\epsilon_{32} = 11111000$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{33} = 00000100$	$\epsilon_{144} = 11110001$	1	4
$\epsilon_{34} = 10000100$	$\epsilon_{206} = 10110011$	1	3
$\epsilon_{35} = 01000100$	$\epsilon_{143} = 01110001$	1	4
$\epsilon_{36} = 11000100$	$\epsilon_{205} = 00110011$	1	4
$\epsilon_{37} = 00100100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{38} = 10100100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{39} = 01100100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{40} = 11100100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{41} = 00010100$	$\epsilon_{144} = 11110001$	1	4
$\epsilon_{42} = 10010100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{43} = 01010100$	$\epsilon_{143} = 01110001$	1	4
$\epsilon_{44} = 11010100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{45} = 00110100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{46} = 10110100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{47} = 01110100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{48} = 11110100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{49} = 00001100$	$\epsilon_{144} = 11110001$	1	4
$\epsilon_{50} = 10001100$	$\epsilon_{206} = 10110011$	1	3
$\epsilon_{51} = 01001100$	$\epsilon_{143} = 01110001$	1	4
$\epsilon_{52} = 11001100$	$\epsilon_{205} = 00110011$	1	4
$\epsilon_{53} = 00101100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{54} = 10101100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{55} = 01101100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{56} = 11101100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{57} = 00011100$	$\epsilon_{144} = 11110001$	1	4
$\epsilon_{58} = 10011100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{59} = 01011100$	$\epsilon_{143} = 01110001$	1	4
$\epsilon_{60} = 11011100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{61} = 00111100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{62} = 10111100$	$\epsilon_{208} = 11110011$	1	4
$\epsilon_{63} = 01111100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{64} = 11111100$	$\epsilon_{207} = 01110011$	1	4
$\epsilon_{65} = 00000010$	$\epsilon_{60} = 11011100$	1	5
$\epsilon_{66} = 10000010$	$\epsilon_{250} = 10011111$	1	3
$\epsilon_{67} = 01000010$	$\epsilon_{59} = 01011100$	1	5
$\epsilon_{68} = 11000010$	$\epsilon_{249} = 00011111$	1	3
$\epsilon_{69} = 00100010$	$\epsilon_{124} = 11011110$	1	3
$\epsilon_{70} = 10100010$	$\epsilon_{252} = 11011111$	1	2
$\epsilon_{71} = 01100010$	$\epsilon_{123} = 01011110$	1	3

$\epsilon_{72} = 11100010$	$\epsilon_{251} = 01011111$	1	3
$\epsilon_{73} = 00010010$	$\epsilon_{60} = 11011100$	1	5
$\epsilon_{74} = 10010010$	$\epsilon_{252} = 11011111$	1	2
$\epsilon_{75} = 01010010$	$\epsilon_{59} = 01011100$	1	5
$\epsilon_{76} = 11010010$	$\epsilon_{251} = 01011111$	1	3
$\epsilon_{77} = 00110010$	$\epsilon_{124} = 11011110$	1	3
$\epsilon_{78} = 10110010$	$\epsilon_{252} = 11011111$	1	2
$\epsilon_{79} = 01110010$	$\epsilon_{123} = 01011110$	1	3
$\epsilon_{80} = 11110010$	$\epsilon_{251} = 01011111$	1	3
$\epsilon_{81} = 00001010$	$\epsilon_{64} = 11111100$	1	5
$\epsilon_{82} = 10001010$	$\epsilon_{254} = 10111111$	1	3
$\epsilon_{83} = 01001010$	$\epsilon_{63} = 01111100$	1	5
$\epsilon_{84} = 11001010$	$\epsilon_{253} = 00111111$	1	3
$\epsilon_{85} = 00101010$	$\epsilon_{128} = 11111110$	1	3
$\epsilon_{86} = 10101010$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{87} = 01101010$	$\epsilon_{127} = 01111110$	1	3
$\epsilon_{88} = 11101010$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{89} = 00011010$	$\epsilon_{64} = 11111100$	1	5
$\epsilon_{90} = 10011010$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{91} = 01011010$	$\epsilon_{63} = 01111100$	1	5
$\epsilon_{92} = 11011010$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{93} = 00111010$	$\epsilon_{128} = 11111110$	1	3
$\epsilon_{94} = 10111010$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{95} = 01111010$	$\epsilon_{127} = 01111110$	1	3
$\epsilon_{96} = 11111010$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{97} = 00000110$	$\epsilon_{192} = 11111101$	1	2
$\epsilon_{98} = 10000110$	$\epsilon_{254} = 10111111$	1	3
$\epsilon_{99} = 01000110$	$\epsilon_{191} = 01111101$	1	2
$\epsilon_{100} = 11000110$	$\epsilon_{253} = 00111111$	1	3
$\epsilon_{101} = 00100110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{102} = 10100110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{103} = 01100110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{104} = 11100110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{105} = 00010110$	$\epsilon_{192} = 11111101$	1	2
$\epsilon_{106} = 10010110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{107} = 01010110$	$\epsilon_{191} = 01111101$	1	2
$\epsilon_{108} = 11010110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{109} = 00110110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{110} = 10110110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{111} = 01110110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{112} = 11110110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{113} = 00001110$	$\epsilon_{192} = 11111101$	1	2
$\epsilon_{114} = 10001110$	$\epsilon_{254} = 10111111$	1	3

$\epsilon_{115} = 01001110$	$\epsilon_{191} = 01111101$	1	2
$\epsilon_{116} = 11001110$	$\epsilon_{253} = 00111111$	1	3
$\epsilon_{117} = 00101110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{118} = 10101110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{119} = 01101110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{120} = 11101110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{121} = 00011110$	$\epsilon_{192} = 11111101$	1	2
$\epsilon_{122} = 10011110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{123} = 01011110$	$\epsilon_{191} = 01111101$	1	2
$\epsilon_{124} = 11011110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{125} = 00111110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{126} = 10111110$	$\epsilon_{256} = 11111111$	1	2
$\epsilon_{127} = 01111110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{128} = 11111110$	$\epsilon_{255} = 01111111$	1	2
$\epsilon_{129} = 00000001$	$\epsilon_{36} = 11000100$	1	5
$\epsilon_{130} = 10000001$	$\epsilon_{226} = 10000111$	1	4
$\epsilon_{131} = 01000001$	$\epsilon_{35} = 01000100$	1	5
$\epsilon_{132} = 11000001$	$\epsilon_{225} = 00000111$	1	3
$\epsilon_{133} = 00100001$	$\epsilon_{100} = 11000110$	1	4
$\epsilon_{134} = 10100001$	$\epsilon_{228} = 11000111$	1	4
$\epsilon_{135} = 01100001$	$\epsilon_{99} = 01000110$	1	3
$\epsilon_{136} = 11100001$	$\epsilon_{227} = 01000111$	1	3
$\epsilon_{137} = 00010001$	$\epsilon_{36} = 11000100$	1	5
$\epsilon_{138} = 10010001$	$\epsilon_{228} = 11000111$	1	4
$\epsilon_{139} = 01010001$	$\epsilon_{35} = 01000100$	1	5
$\epsilon_{140} = 11010001$	$\epsilon_{227} = 01000111$	1	3
$\epsilon_{141} = 00110001$	$\epsilon_{100} = 11000110$	1	4
$\epsilon_{142} = 10110001$	$\epsilon_{228} = 11000111$	1	4
$\epsilon_{143} = 01110001$	$\epsilon_{99} = 01000110$	1	3
$\epsilon_{144} = 11110001$	$\epsilon_{227} = 01000111$	1	3
$\epsilon_{145} = 00001001$	$\epsilon_{48} = 11110100$	1	5
$\epsilon_{146} = 10001001$	$\epsilon_{238} = 10110111$	1	3
$\epsilon_{147} = 01001001$	$\epsilon_{47} = 01110100$	1	5
$\epsilon_{148} = 11001001$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{149} = 00101001$	$\epsilon_{112} = 11110110$	1	3
$\epsilon_{150} = 10101001$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{151} = 01101001$	$\epsilon_{111} = 01110110$	1	3
$\epsilon_{152} = 11101001$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{153} = 00011001$	$\epsilon_{48} = 11110100$	1	5
$\epsilon_{154} = 10011001$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{155} = 01011001$	$\epsilon_{47} = 01110100$	1	5
$\epsilon_{156} = 11011001$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{157} = 00111001$	$\epsilon_{112} = 11110110$	1	3

$\epsilon_{158} = 10111001$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{159} = 01111001$	$\epsilon_{111} = 01110110$	1	3
$\epsilon_{160} = 11111001$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{161} = 00000101$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{162} = 10000101$	$\epsilon_{236} = 10110111$	1	3
$\epsilon_{163} = 01000101$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{164} = 11000101$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{165} = 00100101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{166} = 10100101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{167} = 01100101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{168} = 11100101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{169} = 00010101$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{170} = 10010101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{171} = 01010101$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{172} = 11010101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{173} = 00110101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{174} = 10110101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{175} = 01110101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{176} = 11110101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{177} = 00001101$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{178} = 10001101$	$\epsilon_{236} = 10110111$	1	3
$\epsilon_{179} = 01001101$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{180} = 11001101$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{181} = 00101101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{182} = 10101101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{183} = 01101101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{184} = 11101101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{185} = 00011101$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{186} = 10011101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{187} = 01011101$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{188} = 11011101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{189} = 00111101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{190} = 10111101$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{191} = 01111101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{192} = 11111101$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{193} = 00000011$	$\epsilon_{44} = 11010100$	1	5
$\epsilon_{194} = 10000011$	$\epsilon_{234} = 10010111$	1	3
$\epsilon_{195} = 01000011$	$\epsilon_{43} = 01010100$	1	5
$\epsilon_{196} = 11000011$	$\epsilon_{233} = 00010111$	1	3
$\epsilon_{197} = 00100011$	$\epsilon_{108} = 11010110$	1	3
$\epsilon_{198} = 10100011$	$\epsilon_{236} = 11010111$	1	2
$\epsilon_{199} = 01100011$	$\epsilon_{107} = 01010110$	1	3
$\epsilon_{200} = 11100011$	$\epsilon_{235} = 01010111$	1	3

$\epsilon_{201} = 00010011$	$\epsilon_{44} = 11010100$	1	5
$\epsilon_{202} = 10010011$	$\epsilon_{236} = 11010111$	1	2
$\epsilon_{203} = 01010011$	$\epsilon_{43} = 01010100$	1	5
$\epsilon_{204} = 11010011$	$\epsilon_{235} = 01010111$	1	3
$\epsilon_{205} = 00110011$	$\epsilon_{108} = 11010110$	1	3
$\epsilon_{206} = 10110011$	$\epsilon_{236} = 11010111$	1	2
$\epsilon_{207} = 01110011$	$\epsilon_{107} = 01010110$	1	3
$\epsilon_{208} = 11110011$	$\epsilon_{235} = 01010111$	1	3
$\epsilon_{209} = 00001011$	$\epsilon_{48} = 11110100$	1	5
$\epsilon_{210} = 10001011$	$\epsilon_{238} = 10110111$	1	3
$\epsilon_{211} = 01001011$	$\epsilon_{47} = 01110100$	1	5
$\epsilon_{212} = 11001011$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{213} = 00101011$	$\epsilon_{112} = 11110110$	1	3
$\epsilon_{214} = 10101011$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{215} = 01101011$	$\epsilon_{111} = 01110110$	1	3
$\epsilon_{216} = 11101011$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{217} = 00011011$	$\epsilon_{48} = 11110100$	1	5
$\epsilon_{218} = 10011011$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{219} = 01011011$	$\epsilon_{47} = 01110100$	1	5
$\epsilon_{220} = 11011011$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{221} = 00111011$	$\epsilon_{112} = 11110110$	1	3
$\epsilon_{222} = 10111011$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{223} = 01111011$	$\epsilon_{111} = 01110110$	1	3
$\epsilon_{224} = 11111011$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{225} = 00000111$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{226} = 10000111$	$\epsilon_{238} = 10110111$	1	3
$\epsilon_{227} = 01000111$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{228} = 11000111$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{229} = 00100111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{230} = 10100111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{231} = 01100111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{232} = 11100111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{233} = 00010111$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{234} = 10010111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{235} = 01010111$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{236} = 11010111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{237} = 00110111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{238} = 10110111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{239} = 01110111$	$\epsilon_{239} = 01110111$	1	0
$\epsilon_{240} = 11110111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{241} = 00001111$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{242} = 10001111$	$\epsilon_{238} = 10110111$	1	3
$\epsilon_{243} = 01001111$	$\epsilon_{175} = 01110101$	1	2

$\epsilon_{244} = 11001111$	$\epsilon_{237} = 00110111$	1	3
$\epsilon_{245} = 00101111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{246} = 10101111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{247} = 01101111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{248} = 11101111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{249} = 00011111$	$\epsilon_{176} = 11110101$	1	2
$\epsilon_{250} = 10011111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{251} = 01011111$	$\epsilon_{175} = 01110101$	1	2
$\epsilon_{252} = 11011111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{253} = 00111111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{254} = 10111111$	$\epsilon_{240} = 11110111$	1	2
$\epsilon_{255} = 01111111$	$\epsilon_{239} = 01110111$	1	1
$\epsilon_{256} = 11111111$	$\epsilon_{239} = 01110111$	1	1