UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
CONTROL SYSTEMS ENGINEERING

DETERMINISTIC AND PROBABILISTIC BOOLEAN CONTROL NETWORKS AND THEIR APPLICATION TO GENE REGULATORY NETWORKS

Relatore: Prof. M. Elena Valcher

Laureando: Alberto Chech

ANNO ACCADEMICO 2021 - 2022
Data di laurea 10 ottobre 2022

University of Padova

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

School of Engineering

Department of Information Engineering

Master's Degree in Control Systems Engineering

Master's Thesis

Deterministic and Probabilistic Boolean Control Networks and their application to Gene Regulatory Networks

Supervisor:
Prof. M. Elena Valcher
University of Padova

Master Candidate:
Alberto Chech
2023379

Padova, October 10, 2022

Academic Year 2021-2022

Deterministic and Probabilistic Boolean Control Networks and their application to Gene Regulatory Networks

A tutte le persone a me care, qua e là.

Deterministic and Probabilistic Boolean Control Networks and their application to Gene Regulatory Networks

Alberto Chech

Abstract

This thesis focuses on Deterministic and Probabilistic Boolean Control Networks and their application to some specific Gene Regulatory Networks.
At first, some introductory materials about Boolean Logic and Left Semi-tensor Product are presented in order to explain in detail the concepts of Boolean Networks, Boolean Control Networks, Probabilistic Boolean Networks and Probabilistic Boolean Control Networks. These networks can be modelled in state-space and their representation, obtained by means of the left semi-tensor product, is called algebraic form.
Subsequently, the thesis concentrates on presenting the fundamental properties of these networks such as the classical Systems Theory properties of stability, reachability, controllability and stabilisation. Afterwards, the attention is drawn towards the comparison between deterministic and probabilistic boolean networks.
Finally, two examples of Gene Regulatory Networks are modelled and analysed by means of a Boolean Network and a Probabilistic Boolean Network.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ARE	Anti-oxidant response element
B.C.N.	Boolean Control Network
B.N.	Boolean Network
DNA	Deoxyribonucleic acid
G.A.C.	Globally Attractive Cycle
G.R.N.	Gene Regulatory Network
G.S.	Global Stability
l.c.m.	Least common multiple
L.S.T.P.	Left Semi-Tensor Product
P.B.C.N.	Probabilistic Boolean Control Network
P.B.N.	Probabilistic Boolean Network
PKC	Protein kinase C
ROS	Reactive oxygen species
R.S.T.P.	Right Semi-Tensor Product
SMP	Small Maf proteins
TF	Transcription Factor

Symbols

$\mathcal{B}$	Binary set
$\neg$	Logical negation
$\wedge$	Logical conjunction
$\vee$	Logical disjunction
$\rightarrow$	Logical implication
$\leftrightarrow$	Double logical implication

$\dot{\vee}$	Logical exclusive disjunction
$\mathbb{N}^*$	Collection of the strictly positive integer numbers
$\otimes$	Kronecker tensor product
$\ltimes$	Left semi-tensor product
$\rtimes$	Right semi-tensor product
$\mathbb{R}$	Field of real numbers
$\mathbb{R}^{n\times m}$	Set of all the real matrices of size $n \times m$
$\mathbb{R}^n$	Set of all the real vectors of length n
$\mathrm{Row}_i(A)$	i-th row of matrix A
$\mathrm{Col}_j(A)$	j-th column of matrix A
I_n	Identity matrix of dimension n
δ_n^i	i-th canonical vector of length n
$\mathcal{L}^n$	Set of all the logic vectors of dimension n
$\mathcal{L}^{r\times c}$	Set of all the logic matrices of dimension $r \times c$
$M_\neg$	Structure matrix of logical negation
$M_\wedge$	Structure matrix of logical conjunction
$M_\vee$	Structure matrix of logical disjunction
$W_{[m,n]}$	Swap matrix
M_r	Power reduction matrix
$\mathbf{1}_n$	Unitary column vector of length n
$\emptyset$	Empty set
M_r	Front-maintaining matrix
M_r	Rear-maintaining matrix
$*$	Khatri-Rao matrix product
$M > 0$	Positive matrix
$M \geq 0$	Non-negative matrix
$M \gg 0$	Strictly positive matrix
$\mathcal{I}(\mathbf{G})$	Incidence matrix of graph $\mathbf{G}$
$\mathcal{P}(a)$	Set of the proper factors of a

$\mathbb{Z}$	Field of integer numbers
$\Pr\{E\}$	Probability that event E is true
$\mathbb{N}^{n\times m}$	Set of all the integer matrices of size $n \times m$
ξ_n	Markov chain of dimension n
$\mathrm{Blk}_j(A)$	j-th block of matrix A

Introduction

In this chapter the state of the art of boolean networks and boolean control networks is presented, followed by the description of the structure of the thesis.

State of the Art

A Boolean Network is a dynamic system whose variables are binary and hence described as Boolean, thus representing a model in which the variables are either on and off, high and low, 1 and 0, and the functions that regulate the dynamics are logic functions (combinations of “and”, “or” and “negation” operators). When a Boolean external input is introduced, a Boolean Network becomes a Boolean Control Network. The typical control objective with this model is to find an input sequence such that the system steers from the initial configuration to the desired one.
Boolean Networks have been defined for the first time by Kauffman in 1969 in [1] in order to model genes and their interactions as binary entities. As explained in [2], a gene is a fragment of the DNA that codes one protein, which is the fundamental unit of cellular functions. The main contribution of Kauffman to this theory was to consider genes as either active or inactive and recognizing that genes can activate or inhibit other genes. This idea led to the study of the so-called gene regulatory networks in the biology field. In a gene regulatory network, the role of the activators and inhibitors is very important since they control the patterns of gene expression (active or inactive).
In 2001 in [3] D. Cheng introduced a new operation, called semi-tensor product, which represents a generalization of the classical product between matrices when the dimensions do not coincide. It is interesting to notice that the left-semi tensor product was born for a different purpose and then brilliantly employed in the framework of Boolean Control Networks.
In fact, in 2009 in the paper [4], D. Cheng et al. showed that the semi-tensor product permits to model Boolean Control Networks as a discrete-time state-space model. The new representation is called algebraic state representation and is based on logical canonical vectors and logical matrices. The new approach allows to study the Networks by means of the tools of Systems Theory. Some of the properties that can be characterized by matrices are reachability, controllability, stabilizability, observability; also problems as optimal control, decoupling and fault detection have been investigated, and others still under development.
The main drawback of the algebraic state representation is its computational complexity. In fact, this representation converts a Boolean Control Network with n state variables and m input variables into an equivalent state-space model with 2^n states and 2^m inputs. Then, it is easy to conclude that the operations in this repre-

sentation will have an exponential complexity. However one can consider that the complexity of the algorithms used directly on the Boolean model without any kind of conversion is also exponential.
In 2007 in [5] Akutsu et al. introduced the idea that a Boolean Network can be topologically represented by a directed graph with a set of nodes and a set of edges. Moreover, the authors showed that the algorithms regarding Boolean Networks are NP-hard in general. In the special case in which the network has a tree structure, the control problem can be solved in polynomial time. The other drawback of this representation is that it does not consider the uncertainty of the models. In fact, a Gene Regulatory Network intrinsically contains uncertainty, both in the data and in the model selection.
In 2002 in [6], Shmulevich et al. proposed a new model class, called Probabilistic Boolean Networks, which shares the properties of Boolean Networks but is able to cope with uncertainty. Thus, the new model overcomes the limitation of Boolean Networks which is their inherent determinism. The model can be considered as an interface between the absolute determinism of Boolean Networks and the probabilistic nature of Bayesian Networks. Shmulevich also introduced the concepts of attractors and basin of attraction. The basic idea behind the probabilistic approach is to extend the Boolean Network to accommodate more than one possible function for each node and then to assign a probability to each function.
In 2010 in [7] Cheng et al. studied the theoretical framework of the state-space approach for Boolean Control Networks. The authors showed that a Boolean Network, a logical dynamic system, is only formally the same as the conventional dynamic system. In fact, they differ in the vector space structure. In this paper the concepts of state space, subspace, coordinate transformation, regular subspace and invariant subspace for Boolean Networks are presented.
In recent years, optimal control problems, in which there is a tradeoff between the target and a cost to achieve the target, have been studied in [8]. In 2021 in [9] Cheng studied the dual space of Boolean Networks identifying a dual network, a dual attractor and an invariant subspace. Moreover, the author compared the attractor with the dual attractor and the order of the network with its hidden order.

Chapters Presentation

This thesis is divided into two parts. The first part, consisting of five chapters, treats the theoretical contents of Boolean Control Networks that are needed in the second part, consisting of three chapters, where real cases of Gene Regulatory Networks are tested by means of this theory.
The first chapter explains the preliminary material that is necessary to introduce the concepts of boolean network and boolean control networks. The basis of boolean logic is introduced. This includes the definitions of logic variables, logic functions and truth tables. Then, the tool that permits to build the theory of boolean networks, called left semi-tensor product, is presented along with its properties and some examples. Connecting the last two concepts it is possible to introduce logic vectors and logic matrices, which are the algebraic form of boolean variables and functions. Subsequently, logic functions are united to form logic equations and logic systems. A method to find the algebraic form of these structures is presented and it coincides with the introduction of the structure matrices. Finally, the chapter

contains a brief presentation of non-negative matrices.
The second chapter introduces the boolean networks as dynamical systems of logic variables and functions. Their algebraic form is discussed and a particular graphical representation is defined both for the network context and the state context. Then, the important concepts of fixed points and limit cycles are introduced together with methods to characterize boolean networks. The tools to do a temporal analysis of the convergence of a network are presented. Subsequently, the concepts of basins of attraction and communication classes are explained. Finally, this chapter discusses the stability of boolean networks using the tools previously defined.
In the third chapter, boolean control networks are presented. Since inputs are acting on these systems, it is possible to treat them as switched systems. A new matrix containing the information of the network is defined and a different graphical representation is proposed. The concept of communication classes and the classical Systems Theory concepts of reachability, controllability and stabilisation are treated. Finally, an algorithm to obtain a feedback control matrix is presented in order to stabilise the network.
The fourth and fifth chapters are dedicated to the study of the probabilistic networks. These kind of networks takes into account the possibility that a node is updated at each time by a function randomly selected from a set of functions. In the fourth chapter, probabilistic boolean networks are presented. Since there can be multiple update functions for each state, it is shown that a probabilistic boolean network can be treated as a switched system of multiple boolean networks. The algebraic form and the graphical representation are explained. Then, the concepts of fixed points and limit cycles are adapted to the probabilistic setting. Finally, the stability of a probabilistic boolean network is analysed. In the fifth chapter, probabilistic boolean control networks are presented. Also these networks can be interpreted as switched systems of multiple probabilistic boolean networks. The algebraic form and the graphical representation are explained. Finally, the classical Systems Theory concepts of reachability, controllability and stabilisation with feedback control are treated in the probabilistic setting.
This concludes the theoretical part of the thesis. The second part, consisting of three chapters, is dedicated to the applications of this theory to the Gene Regulatory Networks.
The sixth chapter introduces the concept of Gene Regulatory Networks (GRNs), i.e. biochemical networks that involve genes and proteins, and explains how boolean networks can tackle the problem of representing and analysing GRNs.
The seventh chapter is dedicated to the first application of this thesis: the cellular oxidative stress response. The general scheme of stress response pathways is mentioned, while the oxidative stress response pathways are thoroughly explained since the involved genes and proteins are fundamental to construct the corresponding boolean network model. Finally, some simulations are run to verify that this model is consistent with the original one.
In the eighth chapter, the second and last application is presented: early detection of cancer. Two probabilistic models are built starting from the dynamics of twelve genes in tumor and non-tumor cells. This chapter focuses on testing the consistency of the models with the original systems and compares them both in the logic and algebraic form.

Part I

Theory

Chapter 1

Preliminary Material

This chapter contains the basic knowledge to understand the Boolean Control Networks framework.
The first section recalls what is the Boolean logic and logical variables, operators and functions are presented.
Then, a relatively new matrix product called left semi-tensor product is introduced together with its properties. The product is fundamental in order to transform a system of logic equations in its algebraic form, as it will be shown in the third section.

1.1 Boolean Logic

In Boolean logic the values of the variables are logical values, that is 1 and 0, true and false. A logic variable is defined as follows.

Definition 1.1.1 (Logic Variable)
Consider the binary set $\mathcal{B} = \{0, 1\}$. χ is a logic (or boolean) variable if $\chi \in \mathcal{B}$. It is assumed that $\chi = 1$ corresponds to "true" while $\chi = 0$ corresponds to "false".

Logic variables can be combined using fundamental logic operators such as *NOT, AND, OR*. The effect of each operator on the logic variables can be displayed in a truth table, which shows the output value for all the possible input combinations.

- *NOT*: it is the logical negation operator (also called logical complement) and it is represented with the symbol $\neg$:

$$\begin{aligned} \neg(\cdot) : \mathcal{B} &\to \mathcal{B} \\ \chi &\mapsto \neg\chi. \end{aligned} \tag{1.1.1}$$

  If $\chi = 0$, then $\neg\chi = 1$ and vice-versa because $\neg$ transforms the logical variable into its complement. The truth table is reported in Tab. 1.1.1.

  It is called unary operator since it operates on a single input.
  Negation has the following property:

  - double negation:

$$\neg\neg\chi = \chi \tag{1.1.2}$$

χ	$\neg\chi$
0	1
1	0

Table 1.1.1: Truth Table of Negation (*NOT*)

- *AND*: it is the logical conjunction operator and it is represented with the symbol $\wedge$:

$$\begin{aligned} \wedge(\cdot) : \mathcal{B} \times \mathcal{B} &\to \mathcal{B} \\ (\chi_1, \chi_2) &\mapsto \chi_1 \wedge \chi_2. \end{aligned} \tag{1.1.3}$$

  Only if $\chi_1 = \chi_2 = 1$, then the resulting $\chi_1 \wedge \chi_2 = 1$. The truth table is reported in Tab. 1.1.2.

χ_1	χ_2	$\chi_1 \wedge \chi_2$
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.1.2: Truth Table of Conjunction (*AND*)

  It is called binary operator since it operates on two inputs.
  Conjunction has the following properties:

  - commutativity:
$$\chi_1 \wedge \chi_2 = \chi_2 \wedge \chi_1; \tag{1.1.4}$$
  - associativity:
$$(\chi_1 \wedge \chi_2) \wedge (\chi_3 \wedge \chi_4) = \chi_1 \wedge (\chi_2 \wedge \chi_3) \wedge \chi_4; \tag{1.1.5}$$
  - distributivity with respect to (w.r.t.) $\vee$:
$$(\chi_1 \wedge \chi_2) \vee (\chi_3 \wedge \chi_4) = (\chi_1 \vee \chi_3) \wedge (\chi_1 \vee \chi_4) \wedge (\chi_2 \vee \chi_3) \wedge (\chi_2 \vee \chi_4). \tag{1.1.6}$$

- *OR*: it is the logical disjunction operator and it is represented with the symbol $\vee$:

$$\begin{aligned} \vee(\cdot) : \mathcal{B} \times \mathcal{B} &\to \mathcal{B} \\ (\chi_1, \chi_2) &\mapsto \chi_1 \vee \chi_2. \end{aligned} \tag{1.1.7}$$

  Only if $\chi_1 = \chi_2 = 0$, then the resulting $\chi_1 \vee \chi_2 = 0$. The truth table is reported in Tab. 1.1.3.

  It is called binary operator since it operates on two inputs.
  Conjunction has the following properties:

χ_1	χ_2	$\chi_1 \vee \chi_2$
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.1.3: Truth Table of Disjunction (*OR*)

- commutativity:

$$\chi_1 \vee \chi_2 = \chi_2 \wedge \chi_1; \tag{1.1.8}$$

- associativity:

$$(\chi_1 \vee \chi_2) \vee (\chi_3 \vee \chi_4) = \chi_1 \vee (\chi_2 \vee \chi_3) \vee \chi_4; \tag{1.1.9}$$

- distributivity w.r.t. $\wedge$:

$$(\chi_1 \vee \chi_2) \wedge (\chi_3 \vee \chi_4) = (\chi_1 \wedge \chi_3) \vee (\chi_1 \wedge \chi_4) \vee (\chi_2 \wedge \chi_3) \vee (\chi_2 \wedge \chi_4). \tag{1.1.10}$$

Some basic results are gathered in the following example.

Example 1.1.1 (Fundamental Results)
Given $\chi \in \mathcal{B}$, the following results hold.

$$\begin{aligned}
\chi \wedge 1 &= \chi \\
\chi \vee 1 &= 1 \\
\chi \wedge 0 &= 0 \\
\chi \vee 0 &= \chi \\
\chi \wedge \chi &= \chi \\
\chi \vee \chi &= \chi \\
\chi \wedge \neg\chi &= 0 \\
\chi \vee \neg\chi &= 1.
\end{aligned}$$

From these fundamental operators it is possible to express other derived logical operators. Some examples are reported.

- Implication: it is indicated with $\rightarrow$ and it is defined as follows:

  $$\chi_1 \rightarrow \chi_2 \Leftrightarrow \neg\chi_1 \vee \chi_2. \tag{1.1.11}$$

  The truth table is reported in Tab. 1.1.4.

- Double Implication (or logical equivalence): it is indicated with $\leftrightarrow$ and it is defined as follows:

  $$\chi_1 \leftrightarrow \chi_2 \Leftrightarrow (\chi_1 \rightarrow \chi_2) \wedge (\chi_2 \rightarrow \chi_1) \Leftrightarrow (\neg\chi_1 \vee \chi_2) \wedge (\chi_1 \vee \neg\chi_2). \tag{1.1.12}$$

  The truth table is reported in Tab. 1.1.5.

χ_1	χ_2	$\chi_1 \rightarrow \chi_2$
0	0	1
0	1	1
1	0	0
1	1	1

Table 1.1.4: Truth Table of Implication

χ_1	χ_2	$\chi_1 \leftrightarrow \chi_2$
0	0	1
0	1	0
1	0	0
1	1	1

Table 1.1.5: Truth Table of Double Implication

- *XOR* (or logical exclusive disjunction): it is indicated with $\dot{\vee}$ and it is defined as follows:
$$\chi_1 \dot{\vee} \chi_2 \Leftrightarrow (\neg\chi_1 \wedge \chi_2) \vee (\chi_1 \wedge \neg\chi_2). \tag{1.1.13}$$
The truth table is reported in Tab. 1.1.6.

Then, boolean logic is based on the interaction of boolean variables through boolean operators. It is now possible to define logic functions.

Definition 1.1.2 (Logic Function)
A logic function $f(\cdot)$ is a logical expression that involves $n \in \mathbb{N}^*$ logical variables, such as $\chi_1, \chi_2, \dots, \chi_n \in \mathcal{B}$, which are connected by logical operators. It is a n-ary map:

$$\begin{aligned} f(\cdot) : \mathcal{B}^n &\rightarrow \mathcal{B} \\ (\chi_1, \chi_2, \dots, \chi_n) &\mapsto f(\chi_1, \chi_2, \dots, \chi_n). \end{aligned} \tag{1.1.14}$$

It is clear that all the aforementioned fundamental and derived operators are examples of boolean functions.
A logic function $f : \mathcal{B}^n \rightarrow \mathcal{B}$ with a finite number of boolean variables $n \in \mathbb{N}^*$ has a finite number of input-output combinations. This means that a logic function can be completely represented by means of a proper truth table. The latter will consist of 2^n rows since these corresponds to the number of possible combinations of the inputs. An example is now provided.

Example 1.1.2 (Truth Table of a Logic Function)
Consider the three boolean variables $\chi_1, \chi_2, \chi_3 \in \mathcal{B}$ and the following logic function:

$$f(\chi_1, \chi_2, \chi_3) = (\neg\chi_1 \wedge \chi_2) \vee \chi_3. \tag{1.1.15}$$

The corresponding truth table is reported in Tab. 1.1.7, along with some intermediate steps.

χ_1	χ_2	$\chi_1 \dot{\vee} \chi_2$
0	0	0
0	1	1
1	0	1
1	1	0

Table 1.1.6: Truth Table of Exclusive Disjunction (*XOR*)

χ_1	χ_2	χ_3	$\neg\chi_1$	$\neg\chi_1 \wedge \chi_2$	$(\neg\chi_1 \wedge \chi_2) \vee \chi_3$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	0	0	1

Table 1.1.7: Truth Table of a Logic Function

1.2 Left Semi-tensor Product

The conventional matrix product requires that the number of columns of the first matrix is equal to the number of rows of the second matrix. In order to enable this product even when the matrices dimensions do not match, a generalization is needed. This concept is represented in the literature by the semi-tensor products: the left semi-tensor product (L.S.T.P.), denoted by $\ltimes$, and the right semi-tensor product (R.S.T.P.), denoted by $\rtimes$. In the following only the L.S.T.P. will be presented, since it is the only one used when modelling Boolean Networks.
While in the standard product the entries of the resulting matrix are obtained as a sum of products of the single entries, appearing in row vectors and column vectors, and the Kronecker product (also known as tensor product) corresponds to a multiplication between one entry and a matrix, the semi-tensor product mixes both operations in order to make the dimensions compatible.
First, the Kronecker product is presented.

Definition 1.2.1 (Kronecker Tensor Product)
Given two matrices $A \in \mathbb{R}^{r_1 \times c_1}$ and $B \in \mathbb{R}^{r_2 \times c_2}$ with $r_1, c_1, r_2, c_2 \in \mathbb{N}^*$, the Kronecker product between A and B is defined as:

$$C = A \otimes B := \begin{bmatrix} a_{1,1}B & \cdots & a_{1,c_1}B \\ \vdots & \ddots & \vdots \\ a_{r_1,1}B & \cdots & a_{r_1,c_1}B \end{bmatrix}, \tag{1.2.1}$$

where $a_{i,j}$ represents the entry of matrix A in the i-th row and in the j-th column, with $0 < i \leq r_1$ and $0 < j \leq c_1$. The resulting matrix C has dimensions $r_1 r_2 \times c_1 c_2$.

An example is provided for clarity.

Example 1.2.1 (Kronecker Product)
Let $A = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & 5 \end{bmatrix} \in \mathbb{R}^{2\times 3}$ and $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 2}$. Then, the Kronecker product between A and B is:

$$C = A \otimes B = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 3 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 2 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 5 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 & 0 & 3 \\ 0 & 0 & 2 & 0 & 5 & 0 \\ 0 & 0 & 0 & 2 & 0 & 5 \end{bmatrix}. \quad (1.2.2)$$

The dimensions of the resulting matrix C are 4×6, in accordance with the Def. 1.2.1.

The properties of the Kronecker product are listed in the following proposition.

Proposition 1.2.1 (Kronecker Product Properties)
Let $A \in \mathbb{R}^{r_1 \times c_1}$, $B \in \mathbb{R}^{r_2 \times c_2}$, $C \in \mathbb{R}^{r_3 \times c_3}$ and $D \in \mathbb{R}^{r_4 \times c_4}$ be four matrices with $r_i, c_i \in \mathbb{N}^*$,$\forall i \in \{1, 4\}$. The Kronecker product has the following properties:

1. commutativity with scalars:
$$(\alpha A) \otimes (\beta B) = \alpha\beta(A \otimes B), \quad \alpha, \beta \in \mathbb{R}; \quad (1.2.3)$$
2. associativity:
$$(A \otimes B) \otimes (C \otimes D) = A \otimes (B \otimes C) \otimes D; \quad (1.2.4)$$
3. distributivity w.r.t. addition:
$$(A + B) \otimes (C + D) = (A \otimes C) + (A \otimes D) + (B \otimes C) + (B \otimes D), \quad (1.2.5)$$
if A has the same dimension as B and C has the same dimension as D.

In order to introduce the left semi-tensor product in the best way, two specific simple cases are presented before the general case.

Definition 1.2.2 (Left Semi-tensor Product for Vectors)
Consider the row vector $X = \begin{bmatrix} x_1 & x_2 & \dots & x_m \end{bmatrix} \in \mathbb{R}^{1\times m}$ and the column vector $Y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix}^T \in \mathbb{R}^n$. Let us assume that the length of the row vector is k times the length of the column vector, i.e. $m = kn$, with $m, n, k \in \mathbb{N}^*$. Then, X can be partitioned into n blocks of length k:

$$X = \underbrace{\begin{bmatrix} \underbrace{X^{[1]}}_{k} & \cdots & \underbrace{X^{[n]}}_{k} \end{bmatrix}}_{m=kn}, \quad (1.2.6)$$

where $X^{[i]} \in \mathbb{R}^{1\times k}$ is the i-th block of X, $\forall i = 1, \dots, n$. Then, the L.S.T.P. corresponds to:

$$X \ltimes Y := \sum_{i=1}^{n} X^{[i]} y_i \in \mathbb{R}^{1\times k}. \quad (1.2.7)$$

In the opposite case, i.e. when $n = km$, in a symmetric way, Y can be partitioned into m blocks of length k:

$$Y = \left.\begin{bmatrix} Y^{[1]}\} \, k \\ \vdots \\ Y^{[m]}\} \, k \end{bmatrix}\right\} n = km, \tag{1.2.8}$$

where $Y^{[i]} \in \mathbb{R}^k$ is the i-th block of X, $\forall i = 1, \dots, m$. The L.S.T.P. corresponds to:

$$X \ltimes Y := \sum_{i=1}^{m} x_i Y^{[i]} \ \in \mathbb{R}^k. \tag{1.2.9}$$

Two examples are now presented in the following.

Example 1.2.2 (Left Semi-tensor Product for Vectors)
Consider $X = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{1\times 4}$ and $Y = \begin{bmatrix} 3 & 1 \end{bmatrix}^T \in \mathbb{R}^2$, then $m = 4$, $n = 2$, $k = \frac{m}{n} = 2$. It is possible to recognize $X^{[1]} = \begin{bmatrix} 1 & 2 \end{bmatrix}$ and $X^{[2]} = \begin{bmatrix} 0 & 1 \end{bmatrix}$, $y_1 = 3$ and $y_2 = 1$. Now, by Def. 1.2.2:

$$X \ltimes Y = \begin{bmatrix} 1 & 2 \end{bmatrix} \cdot 3 + \begin{bmatrix} 0 & 1 \end{bmatrix} \cdot 1 = \begin{bmatrix} 3 & 7 \end{bmatrix} \ \in \mathbb{R}^{1\times 2}. \tag{1.2.10}$$

Example 1.2.3 (Left Semi-tensor Product for Vectors)
Consider $X = \begin{bmatrix} 3 & 1 \end{bmatrix} \in \mathbb{R}^{1\times 2}$ and $Y = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix}^T \in \mathbb{R}^4$, then $m = 2$, $n = 4$, $k = \frac{n}{m} = 2$. It is possible to recognize $Y^{[1]} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ and $Y^{[2]} = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$, $x_1 = 3$ and $x_2 = 1$. Now, by Def. 1.2.2:

$$X \ltimes Y = 3 \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 7 \end{bmatrix} \ \in \mathbb{R}^2. \tag{1.2.11}$$

Let us now consider a more general case where the L.S.T.P. is between two matrices.

Definition 1.2.3 (Left Semi-tensor Product for Matrices)
$A \in \mathbb{R}^{r_1 \times c_1}$ and $B \in \mathbb{R}^{r_2 \times c_2}$ are two matrices, with $r_1, c_1, r_2, c_2 \in \mathbb{N}^*$. Suppose that $c_1 = kr_2$ or $r_2 = kc_1$ with $k \in \mathbb{N}^*$. Then the entries of $C = A \ltimes B$ are computed as follows:

$$C^{[i,j]} = \mathrm{Row}_i(A) \ltimes \mathrm{Col}_j(B), \tag{1.2.12}$$

where $C^{[i,j]}$ is the block of C in the i-th row and in the j-th column, since $\mathrm{Row}_i(A)$ represents the i-th row of matrix A and $\mathrm{Col}_j(B)$ represents the j-th column of matrix B.
From Def. 1.2.2, if $c_1 = kr_2$, C is the composition of $r_1 \times c_2$ blocks of dimensions $1 \times k$, so that the final dimensions of C are $r_1 \times kc_2$. Instead, if $r_2 = kc_1$, C is composed of $r_1 \times c_2$ blocks of dimensions $k \times 1$, so that C will have dimensions $kr_1 \times c_2$.

Some examples are reported, for clarity.

Example 1.2.4 (Left Semi-tensor Product for Matrices)
Consider $A = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 2 & 3 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 4}$ and $B = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}$. The dimensions are: $r_1 = 2$, $c_1 = 4$, $r_2 = 2$ and $c_2 = 2$. It follows that $k = 2$. Then:

$$\begin{aligned} C = A \ltimes B &= \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{matrix} \cdot 1 + \\ \cdot 1 + \end{matrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} \begin{matrix} \cdot 2 \\ \cdot 2 \end{matrix} & \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \begin{matrix} \cdot 1 + \\ \cdot 1 + \end{matrix} \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} \begin{matrix} \cdot 0 \\ \cdot 0 \end{matrix} \end{bmatrix} = \\ &= \begin{bmatrix} 3 & 8 & 1 & 2 \\ 2 & 5 & 2 & 3 \end{bmatrix} \in \mathbb{R}^{2\times 2\cdot 2}. \end{aligned} \tag{1.2.13}$$

Example 1.2.5 (Left Semi-tensor Product for Matrices)
Consider $A = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}$ and $B = \begin{bmatrix} 1 & 2 & 1 & 3 \\ 2 & 3 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{2\times 4}$. The dimensions are: $r_1 = 2$, $c_1 = 2$, $r_2 = 4$ and $c_2 = 2$. It follows that $k = 2$. Then:

$$C = A \ltimes B = \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} & 1 \cdot \begin{bmatrix} 2 \\ 3 \end{bmatrix} + 1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 2 \cdot \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 0 \cdot \begin{bmatrix} 1 \\ 3 \end{bmatrix} & 2 \cdot \begin{bmatrix} 2 \\ 3 \end{bmatrix} + 0 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 5 & 4 \\ 2 & 4 \\ 4 & 6 \end{bmatrix} \in \mathbb{R}^{2\cdot 2\times 2}. \quad (1.2.14)$$

As previously discussed, the L.S.T.P. is an extension of the standard matrix product. The latter, in fact, is retrieved when $c_1 = r_2$.
In the following definition the L.S.T.P. in the general case is presented, where it is necessary to resort to the Kronecker tensor product of Def. 1.2.1.

Definition 1.2.4 (Left Semi-tensor Product General Case)
Let us consider two matrices $A \in \mathbb{R}^{r_1\times c_1}$ and $B \in \mathbb{R}^{r_2\times c_2}$ with $r_1, c_1, r_2, c_2 \in \mathbb{N}^*$ and let $T = \text{l.c.m.}(c_1, r_2)$. Then, the L.S.T.P. is defined as follows:

$$A \ltimes B := \left(A \otimes I_{\frac{T}{c_1}}\right)\left(B \otimes I_{\frac{T}{r_2}}\right). \quad (1.2.15)$$

The resulting matrix C has dimensions:

$$\left(\frac{r_1 T}{c_1} \times \frac{c_1 T}{c_1}\right) \times \left(\frac{r_2 T}{r_2} \times \frac{c_2 T}{r_2}\right) = \left(\frac{r_1 T}{c_1} \times \frac{c_2 T}{r_2}\right). \quad (1.2.16)$$

An example is now proposed.

Example 1.2.6 (Left Semi-tensor Product General Case)
Let $A = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \in \mathbb{R}^{2\times 2}$ and $B = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 3 & 2 \end{bmatrix} \in \mathbb{R}^{3\times 2}$ with $r_1 = 2$, $c_1 = 2$, $r_2 = 3$ and $c_2 = 2$. Then $T = \text{l.c.m.}(2, 3) = 2 \cdot 3 = 6$. One obtains that:

$$\begin{aligned} C &= A \ltimes B = (A \otimes I_3)(B \otimes I_2) = \\ &= \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 1 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 3 & 0 & 2 & 0 \\ 0 & 3 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 0 & 2 \\ 6 & 1 & 4 & 0 \\ 2 & 6 & 1 & 4 \\ 1 & 6 & 0 & 3 \\ 9 & 1 & 6 & 0 \\ 2 & 9 & 1 & 6 \end{bmatrix} \in \mathbb{R}^{\frac{2\cdot 6}{2}\times\frac{2\cdot 6}{3}}. \end{aligned} \quad (1.2.17)$$

Let us now explore some properties of the L.S.T.P.in the following proposition.

Proposition 1.2.2 (General Properties)
Let $A \in \mathbb{R}^{r_1\times c_1}$, $B \in \mathbb{R}^{r_2\times c_2}$, $C \in \mathbb{R}^{r_3\times c_3}$ and $D \in \mathbb{R}^{r_4\times c_4}$ be four matrices with $r_i, c_i \in \mathbb{N}^*, \forall\, i \in \{1, 4\}$. The L.S.T.P. has the following properties:

1. commutativity with scalars:

$$(\alpha A) \ltimes (\beta B) = \alpha\beta\,(A \ltimes B), \quad \alpha,\, \beta \in \mathbb{R}; \quad (1.2.18)$$

2. associativity:
$$(A \ltimes B) \ltimes (C \ltimes D) = A \ltimes (B \ltimes C) \ltimes D; \tag{1.2.19}$$

3. distributivity w.r.t. addition:
$$(A + B) \ltimes (C + D) = (A \ltimes C) + (A \ltimes D) + (B \ltimes C) + (B \ltimes D), \tag{1.2.20}$$
if A has the same dimension as B and C has the same dimension as D.

4. transposition:
$$A \ltimes B = B^T \ltimes A^T. \tag{1.2.21}$$

In the following proposition, other properties of the L.S.T.P. that hold in special cases are listed.

Proposition 1.2.3 (Properties in Special Cases)
Let $A \in \mathbb{R}^{r_1 \times c_1}$ and $B \in \mathbb{R}^{r_2 \times c_2}$ be two matrices with $r_1, c_1, r_2, c_2 \in \mathbb{N}^*$. The L.S.T.P. has the following properties:

1. if $c_1 = kr_2$, then:
$$A \ltimes B = A\,(B \ltimes I_k); \tag{1.2.22}$$

2. if $r_2 = kc_1$, then:
$$A \ltimes B = (A \ltimes I_k)\,B; \tag{1.2.23}$$

3. if $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$, then:
$$X \ltimes Y = X \otimes Y. \tag{1.2.24}$$

1.3 Algebraic Form of the Boolean Logic

One of the main objects when dealing with Boolean Networks are logic matrices. In the following definition, both logic vectors and matrices are introduced.

Definition 1.3.1 (Logic Vectors and Matrices)
Consider the identity matrix $I_n \in \mathbb{R}^{n \times n}$ of dimension $n \in \mathbb{N}^*$. The i-th column of I_n, δ_n^i, corresponds to the i-th canonical vector. The only entry different from zero and with unitary value is the one in the i-th row. The set of all logic vectors of dimension n is $\mathcal{L}^n \subset \mathcal{B}^n$. It contains all canonical vectors of dimension n, i.e. $\mathcal{L}^n \triangleq \{\delta_n^i : i = 1, \ldots, n\}$.
A vector is a logic vector if only one entry is equal to one and all the others are equal to zero.
A matrix $L \in \mathbb{R}^{r \times c}$, with r and $c \in \mathbb{N}^*$, is a logic matrix if all its columns belong to $\mathcal{L}^r$, i.e. $\mathrm{Col}_i(L) \subset \mathcal{L}^r$, $\forall i = 1, \ldots, c$. Then, the set of all logic matrices of dimension $(r \times c)$ is $\mathcal{L}^{r \times c} \subset \mathcal{B}^{r \times c}$. A logic matrix $L \in \mathcal{L}^{r \times c}$ is represented as follows:
$$L = \begin{bmatrix} \delta_r^{i_1} & \cdots & \delta_c^{i_r} \end{bmatrix} = \delta_r \begin{bmatrix} i_1 & \cdots & i_c \end{bmatrix}, \tag{1.3.1}$$
where $i_j \in \{1, r\}$, $\forall\, j \in \{1, c\}$.

Logic vectors and matrices are very important in the context of the L.S.T.P., as explained in the following two propositions.

Proposition 1.3.1 (Closeness of $\mathcal{L}^{\cdot}$ w.r.t. L.S.T.P.)
Denote by $\mathcal{L}^{\cdot}$ the set of all canonical vectors of finite dimension. The L.S.T.P. of two logic vectors $L_1 \in \mathcal{L}^m \subset \mathcal{L}^{\cdot}$ and $L_2 \in \mathcal{L}^n \subset \mathcal{L}^{\cdot}$ is another logic vector $L_3 = L_1 \ltimes L_2 \in \mathcal{L}^{m \cdot n}$, $\forall\, n,\, m \in \mathbb{N}^*$.

An example is given in the following.

Example 1.3.1 (Closeness of $\mathcal{L}^{\cdot}$ w.r.t. L.S.T.P.)
Let us consider two logic vectors $L_1 = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \in \mathcal{L}^3$ and $L_2 = \begin{bmatrix}0 & 1\end{bmatrix}^T \in \mathcal{L}^2$. The L.S.T.P. is another logic vector as follows:

$$L_3 = L_1 \ltimes L_2 = \begin{bmatrix}0\\1\\0\\0\\0\\0\end{bmatrix} \in \mathcal{L}^{3 \cdot 2} = \mathcal{L}^6. \tag{1.3.2}$$

Proposition 1.3.2 (Closeness of $\mathcal{L}^{\times}$ w.r.t. L.S.T.P.)
Denote by $\mathcal{L}^{\times}$ the set of all logic matrices of finite dimensions. The L.S.T.P. of two logic matrices $A \in \mathcal{L}^{r_1 \times c_1} \subset \mathcal{L}^{\times}$ and $B \in \mathcal{L}^{r_2 \times c_2} \subset \mathcal{L}^{\times}$, with r_1, c_1, r_2 and $c_2 \in \mathbb{N}^*$, is another logic matrix $C = A \ltimes B \in \mathcal{L}^{\frac{r_1 T}{c_1} \times \frac{c_2 T}{r_2}} \subset \mathcal{L}^{\times}$, with $T = \text{l.c.m.}(c_1,\, r_2) \in \mathbb{N}^*$.

An example is given in the following.

Example 1.3.2 (Closeness of $\mathcal{L}^{\times}$ w.r.t. L.S.T.P.)
Let us consider two logic matrices $A = \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{bmatrix} \in \mathcal{L}^{3\times 3}$ and $B = \begin{bmatrix}0 & 1 & 1\\1 & 0 & 0\end{bmatrix} \in \mathcal{L}^{2\times 3}$. The L.S.T.P. is another logic matrix as follows:

$$C = A \ltimes B = \begin{bmatrix}0&0&0&1&0&0&1&0&0\\0&0&0&0&1&0&0&1&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&1\\1&0&0&0&0&0&0&0&0\end{bmatrix} \in \mathcal{L}^{6\times 9}. \tag{1.3.3}$$

With the new knowledge of logic vectors, one can associate bijectively a boolean variable to a vector in $\mathcal{L}^2$. A formal definition is presented next.

Definition 1.3.2 (Boolean Variable and Logic Vector)
There exists a bijective relation between a generic boolean variable $\chi \in \mathcal{B}$ and a logic vector $x \in \mathcal{L}^2$. In fact, it is true that:

$$x \leftrightarrow \begin{bmatrix}\chi\\\neg\chi\end{bmatrix}, \tag{1.3.4}$$

whose realizations are:

$$\begin{cases} 1 \leftrightarrow \delta_2^1 = \begin{bmatrix}1\\0\end{bmatrix} \\ 0 \leftrightarrow \delta_2^2 = \begin{bmatrix}0\\1\end{bmatrix} \end{cases}. \tag{1.3.5}$$

In the next definition it is shown that the bijective correspondence between $\mathcal{B}$ and $\mathcal{L}^2$ can be extended to $\mathcal{B}^n$ and $\mathcal{L}^{2^n}$.

Definition 1.3.3 (From Logic Vectors to Boolean Variables)
Given n logic variables $\chi_i \in \mathcal{B}$, with $i = 1, \ldots, n$ and $n \in \mathbb{N}^*$, consider their vectorial forms $x_i \in \mathcal{L}^2$ as in Def. 1.3.2. Define $\boldsymbol{\chi}$ as follows:

$$\boldsymbol{\chi} = \begin{bmatrix}\chi_1 & \chi_2 & \cdots & \chi_n\end{bmatrix}^T \in \mathcal{B}^n, \tag{1.3.6}$$

and define $\mathbf{x}$ as follows:

$$\mathbf{x} := \ltimes_{i=1}^{n} x_i = x_1 \ltimes x_2 \ltimes \cdots \ltimes x_n \in \mathcal{L}^{2^n}. \tag{1.3.7}$$

It is now possible to extend the bijective correspondence between $\mathcal{B}$ and $\mathcal{L}^2$ to $\mathcal{B}^n$ and $\mathcal{L}^{2^n}$:

$$\begin{aligned}\mathbf{x} &:= \begin{bmatrix}\chi_1\\ \neg\chi_1\end{bmatrix} \ltimes \begin{bmatrix}\chi_2\\ \neg\chi_2\end{bmatrix} \ltimes \cdots \ltimes \begin{bmatrix}\chi_n\\ \neg\chi_n\end{bmatrix} = \\ &= \begin{bmatrix}\chi_1\chi_2\cdots\chi_n\\ \chi_1\chi_2\cdots\neg\chi_n\\ \vdots\\ \neg\chi_1\neg\chi_2\cdots\neg\chi_n\end{bmatrix} \in \mathcal{L}^{2^n}.\end{aligned} \tag{1.3.8}$$

Since it is a bijective correspondence, it means that the information contained in the newly defined logic vector is the same information contained in the boolean variables.
So, given $\mathbf{x} \in \mathcal{L}^{2^n}$ as in Def. 1.3.3, one can reconstruct the n original boolean variables.
The next proposition shows how to move from boolean variables to logic vectors and vice-versa.

Proposition 1.3.3 (Boolean Variables $\Leftrightarrow$ Logic Vectors)
Let $\chi_i \in \mathcal{B}$, with $i \in \{1, n\}$, be the n boolean variables and $\mathbf{x} = \delta_{2^n}^j \in \mathcal{L}^{2^n}$ be the corresponding logic vector, with n and $j \in \mathbb{N}^*$.

- From boolean variables to logic vector: given the set of χ_i's, the value j of $\mathbf{x} = \delta_{2^n}^j$ is retrieved as follows:

$$j = \sum_{i=1}^{n} (1 - \chi_i)\, 2^{n-i} + 1. \tag{1.3.9}$$

- From logic vector to boolean variables: first, define $q_0 \triangleq 2^n - j$. Then, iteratively, compute:

$$\begin{cases}\chi_i = \lfloor \frac{q_{i-1}}{2^{n-i}} \rfloor\\ q_i = q_{i-1} - 2^{n-i}\chi_i\end{cases}, i = 1, \ldots, n. \tag{1.3.10}$$

Two examples are presented to explain both directions.

Example 1.3.3 (From the χ_i's to $\mathbf{x}$)
Given $x_1 = \delta_2^1$, $x_2 = \delta_2^2$, $x_3 = \delta_2^1$ and $x_4 = \delta_2^1$, $\mathbf{x} = \delta_{2^n}^j = x_1 \ltimes x_2 \ltimes x_3 \ltimes x_4$ is determined as follows:

$$\begin{cases} x_1 = \delta_2^1 \to \chi_1 = 1 \\ x_2 = \delta_2^2 \to \chi_2 = 0 \\ x_3 = \delta_2^1 \to \chi_3 = 1 \\ x_4 = \delta_2^1 \to \chi_4 = 1 \\ j = \sum_{i=1}^4 (1-\chi_i)\, 2^{4-i} + 1 = 2^2 + 1 = 5 \to \mathbf{x} = \delta_{16}^5. \end{cases} \tag{1.3.11}$$

Example 1.3.4 (From $\mathbf{x}$ to the χ_i's)
Let $\mathbf{x} = \delta_8^6 = x_1 \ltimes x_2 \ltimes x_3$. Then, $n = 3$, $j = 6$ and $q_0 = 2^3 - 6 = 2$. Following the procedure of Prop. 1.3.3, one gets:

$$\begin{cases} \chi_1 = \lfloor \frac{q_0}{2^2} \rfloor = \lfloor \frac{2}{4} \rfloor = 0 \to x_1 = \delta_2^2 \\ q_1 = q_0 - 2^2 \cdot \chi_1 = 2 - 0 = 2 \\ \chi_2 = \lfloor \frac{2}{2} \rfloor = 1 \to x_2 = \delta_2^1 \\ q_2 = 1 - 1 = 0 \\ \chi_3 = \lfloor \frac{0}{1} \rfloor = 0 \to x_3 = \delta_2^2 \\ q_3 = 0 - 0 = 0. \end{cases} \tag{1.3.12}$$

Since boolean variables can be expressed through logic vectors, the same can be applied to the result of a boolean function. Since it belongs to $\mathcal{B}$ it can be expressed as a vector in $\mathcal{L}^2$. This translates into expressing the original truth table of a logic function in terms of logic vectors. That is, given a boolean function $f(\chi_1, \dots, \chi_n) : \mathcal{B}^n \to \mathcal{B}$, with $\chi_i \in \mathcal{B}, \forall i = \{1, \dots, n\}$ and $n \in \mathbb{N}^*$, it is possible to find its equivalent representation $\hat{f}(\chi_1, \dots, \chi_n) : \mathcal{L}^{2^n} \to \mathcal{L}^2$. The following example shows the conversion to logic vectors in the truth table.

Example 1.3.5 (Truth Table in Vectorial Form)
Using the function of Ex. 1.1.2, the truth table of Tab. 1.1.7 is translated into Tab. 1.3.1.

x_1	x_2	x_3	$\mathbf{x}$	$f(\mathbf{x})$
δ_2^2	δ_2^2	δ_2^2	δ_8^8	δ_2^2
δ_2^2	δ_2^2	δ_2^1	δ_8^7	δ_2^1
δ_2^2	δ_2^1	δ_2^2	δ_8^6	δ_2^1
δ_2^2	δ_2^1	δ_2^1	δ_8^5	δ_2^1
δ_2^1	δ_2^2	δ_2^2	δ_8^4	δ_2^2
δ_2^1	δ_2^2	δ_2^1	δ_8^3	δ_2^1
δ_2^1	δ_2^1	δ_2^2	δ_8^2	δ_2^2
δ_2^1	δ_2^1	δ_2^1	δ_8^1	δ_2^1

Table 1.3.1: Truth Table in Vectorial Form of a Logic Function

Now it is possible to state that a logic function can be expressed in Algebraic Form using logic vectors and matrices.

Theorem 1.3.1 (Algebraic Form of a Logic Function)
Let $f(\chi_1, \ldots, \chi_n) : \mathcal{B}^n \to \mathcal{B}$, with $\chi_i \in \mathcal{B}, \forall i = \{1, \ldots, n\}$ and $n \in \mathbb{N}^*$, be a logic function. The equivalent function $\hat{f}(\chi_1, \ldots, \chi_n) : \mathcal{L}^{2^n} \to \mathcal{L}^2$, with $x_i \in \mathcal{L}^2$, $\forall i = 1, \ldots, n$, can be expressed in Algebraic Form in the following way:

$$\hat{f}(\chi_1, \ldots, \chi_n) = M_f \ltimes \mathbf{x} \in \mathcal{L}^2, \tag{1.3.13}$$

where $\mathbf{x} := \ltimes_{i=1}^{n} x_i \in \mathcal{L}^{2^n}$ and M_f is a logic matrix $\in \mathcal{L}^{2\times 2^n}$ called Structure Matrix of $\hat{f}(\cdot)$.

An example on how to find the structure matrix is given in the following.

Example 1.3.6 (Structure Matrix)
Using the same function $f(\cdot)$ of Ex. 1.1.2, one can compute the vector $\mathbf{x} = x_1 \ltimes x_2 \ltimes x_3$. From the results of the truth table in vectorial form (see Tab. 1.3.1), M_f is defined as follows:

$$\begin{cases} M_f \ltimes \delta_8^1 = \mathrm{Col}_1(M_f) = \delta_2^1 \\ M_f \ltimes \delta_8^2 = \mathrm{Col}_2(M_f) = \delta_2^2 \\ M_f \ltimes \delta_8^3 = \mathrm{Col}_3(M_f) = \delta_2^1 \\ M_f \ltimes \delta_8^4 = \mathrm{Col}_4(M_f) = \delta_2^2 \\ M_f \ltimes \delta_8^5 = \mathrm{Col}_5(M_f) = \delta_2^1 \\ M_f \ltimes \delta_8^6 = \mathrm{Col}_6(M_f) = \delta_2^1 \\ M_f \ltimes \delta_8^7 = \mathrm{Col}_7(M_f) = \delta_2^1 \\ M_f \ltimes \delta_8^8 = \mathrm{Col}_8(M_f) = \delta_2^2. \end{cases} \tag{1.3.14}$$

Therefore:

$$M_f = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \in \mathcal{L}^{2\times 8}. \tag{1.3.15}$$

In the following proposition, the structure matrices of the fundamental operators are presented, since they are logic functions.

Proposition 1.3.4 (Structure Matrices of Fundamental Operators)
Using Thm. 1.3.1, it possible to construct the structure matrices of the fundamental operators in Def. 1.1.1:

- *NOT*: given $x \in \mathcal{L}^2$, the structure matrix for negation is denoted by $M_\neg$ so that $\neg x \in \mathcal{L}^2$ is found as $\neg x = M_\neg \ltimes x$. In fact, looking at Tab. 1.1.1, one gets:

$$\begin{cases} x = \delta_2^1 \to \neg x = M_\neg \ltimes \delta_2^1 = \mathrm{Col}_1(M_\neg) = \delta_2^2 \\ x = \delta_2^2 \to \neg x = M_\neg \ltimes \delta_2^2 = \mathrm{Col}_2(M_\neg) = \delta_2^1, \end{cases} \tag{1.3.16}$$

  and

$$M_\neg = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}; \tag{1.3.17}$$

- *AND*: given $x_1, x_2 \in \mathcal{L}^2$, the structure matrix for conjunction is denoted by $M_\wedge$ so that $x_1 \wedge x_2 \in \mathcal{L}^2$ is found as $x_1 \wedge x_2 = M_\wedge \ltimes x_1 \ltimes x_2$. In fact, looking at Tab. 1.1.2, one gets:

$$\begin{cases} x_1 = x_2 = \delta_2^1 & \to x_1 \wedge x_2 = M_\wedge \ltimes x_1 \ltimes x_2 = \mathrm{Col}_1(M_\wedge) = \delta_2^1 \\ x_1 = \delta_2^1,\, x_2 = \delta_2^2 & \to x_1 \wedge x_2 = M_\wedge \ltimes x_1 \ltimes x_2 = \mathrm{Col}_2(M_\wedge) = \delta_2^2 \\ x_1 = \delta_2^2,\, x_2 = \delta_2^1 & \to x_1 \wedge x_2 = M_\wedge \ltimes x_1 \ltimes x_2 = \mathrm{Col}_3(M_\wedge) = \delta_2^2 \\ x_1 = x_2 = \delta_2^2 & \to x_1 \wedge x_2 = M_\wedge \ltimes x_1 \ltimes x_2 = \mathrm{Col}_4(M_\wedge) = \delta_2^2, \end{cases} \tag{1.3.18}$$

and

$$M_\wedge = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}; \tag{1.3.19}$$

- *OR*: given $x_1,\, x_2 \in \mathcal{L}^2$, the structure matrix for conjunction is denoted by $M_\vee$ so that $x_1 \vee x_2 \in \mathcal{L}^2$ is found as $x_1 \vee x_2 = M_\vee \ltimes x_1 \ltimes x_2$. In fact, looking at Tab. 1.1.3, one gets:

$$\begin{cases} x_1 = x_2 = \delta_2^1 & \to x_1 \vee x_2 = M_\vee \ltimes x_1 \ltimes x_2 = \mathrm{Col}_1(M_\vee) = \delta_2^1 \\ x_1 = \delta_2^1,\, x_2 = \delta_2^2 & \to x_1 \vee x_2 = M_\vee \ltimes x_1 \ltimes x_2 = \mathrm{Col}_2(M_\vee) = \delta_2^1 \\ x_1 = \delta_2^2,\, x_2 = \delta_2^1 & \to x_1 \vee x_2 = M_\vee \ltimes x_1 \ltimes x_2 = \mathrm{Col}_3(M_\vee) = \delta_2^1 \\ x_1 = x_2 = \delta_2^2 & \to x_1 \vee x_2 = M_\vee \ltimes x_1 \ltimes x_2 = \mathrm{Col}_4(M_\vee) = \delta_2^2, \end{cases} \tag{1.3.20}$$

and

$$M_\vee = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.3.21}$$

The L.S.T.P. does not have the commutative property. Nevertheless, it is possible to introduce some particular permutation matrices in order to obtain a pseudo-commutative property. In the context of L.S.T.P., a useful permutation matrix is the swap matrix.

Definition 1.3.4 (Swap Matrix)
A swap matrix $W_{[m,\,n]} \in \mathcal{L}^{mn\times mn}$ is a permutation matrix with this structure:

$$\begin{aligned} W_{[m,\,n]} := \delta_{mn}[&1,\, m+1,\, 2m+1,\, \dots,\, (n-1)\,m+1, \\ &2,\, m+2,\, 2m+2,\, \dots,\, (n-1)\,m+2, \\ &\dots, \\ &m,\, m+m,\, 2m+m,\, \dots,\, (n-1)\,m+m]. \end{aligned} \tag{1.3.22}$$

The use of the swap matrix is clarified in the following proposition.

Proposition 1.3.5 (Pseudo-commutativity)
Consider two vectors $X \in \mathbb{R}^m$, $Y \in \mathbb{R}^n$ and a matrix $A \in \mathbb{R}^{r\times c}$, with $m,\, n,\, r,\, c \in \mathbb{N}^*$. Then the following properties hold:

- vectors exchange:

$$W_{[m,\,n]} \ltimes X \ltimes Y = Y \ltimes X; \tag{1.3.23}$$

- vector-matrix exchange:

$$X \ltimes A = W_{[r,\,m]} \ltimes A \ltimes W_{[m,\,c]} \ltimes X = (I_m \otimes A) \ltimes X. \tag{1.3.24}$$

The dimensions of the above matrices are: $W_{[m,\,n]} \in \mathcal{L}^{mn\times mn}$, $W_{[r,\,m]} \in \mathcal{L}^{rm\times rm}$ and $W_{[m,\,c]} \in \mathcal{L}^{mc\times mc}$.

Let us consider two examples: one for vectors exchange and another for vector-matrix exchange.

Example 1.3.7 (Vectors Exchange)
Assume that $\mathbf{x}' = x_2 \ltimes x_1$, where $x_1, x_2 \in \mathcal{L}^2$. In order to write $\mathbf{x} = x_1 \ltimes x_2$, it is necessary to swap x_1 and x_2. Applying Def. 1.3.4 and observing that $m = n = 2$, one gets that:

$$\begin{aligned} \mathbf{x} &= x_1 \ltimes x_2 = W_{[2,2]} \ltimes \mathbf{x}' = W_{[2,2]} \ltimes x_2 \ltimes x_1 = \\ &= \delta_4 \begin{bmatrix} 1 & 3 & 2 & 4 \end{bmatrix} \ltimes x_2 \ltimes x_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes x_2 \ltimes x_1. \end{aligned} \tag{1.3.25}$$

Example 1.3.8 (Vector-Matrix Exchange)
Assume that $\mathbf{y}' = x_1 \ltimes A \ltimes x_2$, where $x_1, x_2 \in \mathcal{L}^2$, and $A \in \mathbb{R}^{2\times2}$. Applying Def. 1.3.4 and observing that $m = r = c = 2$, one gets $W_{[r,m]} = W_{[m,c]} = W_{[2,2]}$. In order to determine $\mathbf{y} = (I_m \otimes A) \ltimes x_1 \ltimes x_2 = (I_m \otimes A) \ltimes \mathbf{x}$, one finds:

$$\begin{aligned} \mathbf{y} &= (I_m \otimes A) \ltimes \mathbf{x} = (I_m \otimes A) \ltimes x_1 \ltimes x_2 = \\ &= W_{[2,2]} \ltimes A \ltimes W_{[2,2]} \ltimes x_1 \ltimes x_2 = x_1 \ltimes A \ltimes x_2 = \mathbf{y}'. \end{aligned} \tag{1.3.26}$$

In the following proposition, the power reduction matrix M_r is introduced. This operator is important to reduce the second power of a logic vector.

Proposition 1.3.6 (Power Reduction Matrix)
Consider $x \in \mathcal{L}^2$, then the following holds:

$$x^2 = x \ltimes x = \begin{bmatrix} \chi \\ \neg\chi \end{bmatrix} \ltimes \begin{bmatrix} \chi \\ \neg\chi \end{bmatrix} = \begin{bmatrix} \chi\chi \\ \chi\neg\chi \\ \neg\chi\chi \\ \neg\chi\neg\chi \end{bmatrix} = \begin{bmatrix} \chi \\ 0 \\ 0 \\ \chi \end{bmatrix}. \tag{1.3.27}$$

Introducing the power reduction matrix M_r, the equation becomes:

$$x^2 = M_r \ltimes x, \tag{1.3.28}$$

where $M_r = I_4 \begin{bmatrix} 1 & 4 \end{bmatrix}$.

Thanks to the introduction of the structure matrices of the fundamental operators, it is now possible to convert a logic function in its equivalent algebraic form starting from its variables $\in \mathcal{L}^2$.
Here is a simple algorithm to obtain the structure matrix of a logic function:

1. First, all the boolean operators must be substituted with the corresponding structure matrices;
2. then, the objective is to move all variables to the right side of the function and all logic matrices to the left side of the function;
3. then, the logic vectors on the right must be reordered w.r.t. their subscript index, as seen in previous examples;

4. now, if there are variables with order greater than one, they must be reduced with the power reduction matrix M_r;
5. lastly, all matrices M_r of the previous point must be brought to the left of all logic vectors.

An example where to apply this algorithm is now proposed.

Example 1.3.9 (Logic Function in Algebraic Form)
Let us consider the following logic function:

$$f(\chi_1, \chi_2) = (\chi_1 \wedge \chi_2) \rightarrow \chi_2. \tag{1.3.29}$$

Remembering the conversion of implication, the function is rewritten as:

$$f(\chi_1, \chi_2) = (\chi_1 \wedge \chi_2) \rightarrow \chi_2 = \neg(\chi_1 \wedge \chi_2) \vee \chi_2. \tag{1.3.30}$$

Following the points of the algorithm, one gets:

$$\begin{aligned} f(\chi_1, \chi_2) &= \neg(\chi_1 \wedge \chi_2) \vee \chi_2 = \\ &= M_\vee \ltimes M_\neg \ltimes M_\wedge \ltimes x_1 \ltimes x_2 \ltimes x_2 = \\ &= M_\vee \ltimes M_\neg \ltimes M_\wedge \ltimes x_1 \ltimes x_2^2 = \\ &= M_\vee \ltimes M_\neg \ltimes M_\wedge \ltimes x_1 \ltimes M_r \ltimes x_2 = \\ &= M_\vee \ltimes M_\neg \ltimes M_\wedge \ltimes (I_2 \otimes M_r) \ltimes x_1 \ltimes x_2 = \\ &= M_f \ltimes x_1 \ltimes x_2 = \\ &= M_f \ltimes \mathbf{x}, \end{aligned} \tag{1.3.31}$$

where $M_f = M_\vee \ltimes M_\neg \ltimes M_\wedge \ltimes (I_2 \otimes M_r) \in \mathcal{L}^{2\times 4}$ and $\mathbf{x} = x_1 \ltimes x_2 \in \mathcal{L}^2$.

1.4 Logic Equations and Logic Systems

In the following, logic equations and logic systems will be treated.

Definition 1.4.1 (Logic Equation)
A logic equation asserts the equality of two logic expression. It can be expressed as:

$$f(\chi_1, \dots, \chi_n) = \beta, \tag{1.4.1}$$

where $f(\chi_1, \dots, \chi_n)$ is a logic function as defined in Def. 1.1.2 and $\beta \in \mathcal{B}$ is a fixed boolean constant, thus assuming either the value $\beta = 0$ or $\beta = 1$. The equation is satisfied for a set of logic constants $\gamma_1, \dots, \gamma_n \in \mathcal{B}$ such that $\chi_i = \gamma_i$, $\forall i \in \{1, n\}$.

Let us now see an example of logic equation and its solution.

Example 1.4.1 (Logic Equation)
Consider the following logic equation:

$$f(\chi_1, \chi_2, \chi_3) = (\neg\chi_1 \wedge \chi_2) \vee \chi_3 = \beta, \tag{1.4.2}$$

with χ_1, χ_2, χ_3 and $\beta \in \mathcal{B}$.
In order to find the solution of the equation, one needs to find the inputs combinations such that the output of the logic function matches each value of β. Let us call the variables vector $\boldsymbol{\chi} = \begin{bmatrix}\chi_1 & \chi_2 & \chi_3\end{bmatrix}^T$ and the solution vector $\boldsymbol{\gamma} = \begin{bmatrix}\gamma_1 & \gamma_2 & \gamma_3\end{bmatrix}^T$. Looking at the truth table in Tab. 1.1.7, the solutions can be found as follows:

- for $\beta = 0$:

$$\begin{cases} \boldsymbol{\chi} = \gamma_1^0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_2^0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_3^0 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}^T, \end{cases} \tag{1.4.3}$$

  the solution set is:

$$\boldsymbol{\chi} = \gamma^0 = \left\{\gamma_1^0,\, \gamma_2^0,\, \gamma_3^0\right\}; \tag{1.4.4}$$

- for $\beta = 1$:

$$\begin{cases} \boldsymbol{\chi} = \gamma_1^1 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_2^1 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_3^1 = \begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_4^1 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_5^1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T, \end{cases} \tag{1.4.5}$$

  the solution set is:

$$\boldsymbol{\chi} = \gamma^1 = \left\{\gamma_1^1,\, \gamma_2^1,\, \gamma_3^1,\, \gamma_4^1,\, \gamma_5^1\right\}. \tag{1.4.6}$$

Definition 1.4.2 (Logic System)
A logic system is a collection of $m \in \mathbb{N}^*$ logic equations with $f_1(\cdot)$, …, $f_m(\cdot)$, logic functions, whose variables are $\chi_i \in \mathcal{B}$, with $i = 1, \ldots, n$, and $\beta_1, \ldots, \beta_m \in \mathcal{B}$. It can be expressed as:

$$\begin{cases} f_1(\chi_{1_1}, \ldots, \chi_{n_1}) = \beta_1 \\ f_2(\chi_{1_2}, \ldots, \chi_{n_2}) = \beta_2 \\ \vdots \\ f_m(\chi_{1_m}, \ldots, \chi_{n_m}) = \beta_m, \end{cases} \tag{1.4.7}$$

where the subscript i_j indicates that the i-th variable is involved in the j-th equation, with $0 \le i_j \le n_j$, $0 \le n_j \le n$ and $0 < j \le m$. In this case, the system solution is a set of boolean constants $\gamma_1, \ldots, \gamma_n \in \mathcal{B}$ which satisfies all the equations in the system at the same time.

Let us now see an example of logic system and its solution.

Example 1.4.2 (Logic System)
Consider the following logic system:

$$\begin{cases} f(\chi_1, \chi_2) = \neg\chi_1 \vee \chi_2 = \beta_1 \\ f(\chi_2, \chi_3) = \chi_2 \wedge \neg\chi_3 = \beta_2. \end{cases} \tag{1.4.8}$$

with χ_1, χ_2, χ_3, β_1 and $\beta_2 \in \mathcal{B}$.
In order to find the solution of the system, $\boldsymbol{\chi} = \begin{bmatrix} \chi_1 & \chi_2 & \chi_3 \end{bmatrix}^T = \begin{bmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{bmatrix}^T = \boldsymbol{\gamma}$, for each combination of the constant values $\boldsymbol{\beta} = \begin{bmatrix} \beta_1 & \beta_2 \end{bmatrix}^T$, one needs to find the solutions for both logic functions. Looking at the truth table in Tab. 1.4.1, the solutions can be found as follows:

- for $\beta = \begin{bmatrix}0 & 0\end{bmatrix}^T$:

$$\begin{cases} \boldsymbol{\chi} = \gamma_1^{[0,0]} = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_2^{[0,0]} = \begin{bmatrix}1 & 0 & 1\end{bmatrix}^T , \end{cases} \tag{1.4.9}$$

the solution set is:

$$\boldsymbol{\chi} = \gamma^{[0,0]} = \left\{\gamma_1^{[0,0]}, \gamma_2^{[0,0]}\right\} ; \tag{1.4.10}$$

- for $\beta = \begin{bmatrix}0 & 1\end{bmatrix}^T$, there are no acceptable solutions:

$$\boldsymbol{\chi} = \gamma^{[0,1]} = \{\emptyset\} ; \tag{1.4.11}$$

- for $\beta = \begin{bmatrix}1 & 0\end{bmatrix}^T$:

$$\begin{cases} \boldsymbol{\chi} = \gamma_1^{[1,0]} = \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_2^{[1,0]} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_3^{[1,0]} = \begin{bmatrix}0 & 1 & 1\end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_4^{[1,0]} = \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T , \end{cases} \tag{1.4.12}$$

the solution set is:

$$\boldsymbol{\chi} = \gamma^{[1,0]} = \left\{\gamma_1^{[1,0]}, \gamma_2^{[1,0]}, \gamma_3^{[1,0]}, \gamma_4^{[1,0]}\right\} ; \tag{1.4.13}$$

- for $\beta = \begin{bmatrix}1 & 1\end{bmatrix}^T$:

$$\begin{cases} \boldsymbol{\chi} = \gamma_1^{[1,1]} = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T \\ \boldsymbol{\chi} = \gamma_2^{[1,1]} = \begin{bmatrix}1 & 1 & 0\end{bmatrix}^T , \end{cases} \tag{1.4.14}$$

the solution set is:

$$\boldsymbol{\chi} = \gamma^{[1,1]} = \left\{\gamma_1^{[1,1]}, \gamma_2^{[1,1]}\right\} . \tag{1.4.15}$$

χ_1	χ_2	χ_3	$\boldsymbol{\chi} = \begin{bmatrix}\chi_1 & \chi_2 & \chi_3\end{bmatrix}^T$	$f_1 = \neg\chi_1 \vee \chi_2$	$f_2 = \chi_2 \wedge \neg\chi_3$	$\mathbf{f} = \begin{bmatrix}f_1 & f_2\end{bmatrix}^T$
0	0	0	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$	1	0	$\begin{bmatrix}1 & 0\end{bmatrix}^T$
0	0	1	$\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$	1	0	$\begin{bmatrix}1 & 0\end{bmatrix}^T$
0	1	0	$\begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$	1	1	$\begin{bmatrix}1 & 1\end{bmatrix}^T$
0	1	1	$\begin{bmatrix}0 & 1 & 1\end{bmatrix}^T$	1	0	$\begin{bmatrix}1 & 0\end{bmatrix}^T$
1	0	0	$\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$	0	0	$\begin{bmatrix}0 & 0\end{bmatrix}^T$
1	0	1	$\begin{bmatrix}1 & 0 & 1\end{bmatrix}^T$	0	0	$\begin{bmatrix}0 & 0\end{bmatrix}^T$
1	1	0	$\begin{bmatrix}1 & 1 & 0\end{bmatrix}^T$	1	1	$\begin{bmatrix}1 & 1\end{bmatrix}^T$
1	1	1	$\begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$	1	0	$\begin{bmatrix}1 & 0\end{bmatrix}^T$

Table 1.4.1: Truth Table of a Logic System

At this point it is interesting to understand how to convert logic equations and systems to an equivalent algebraic form. This is the topic of the next theorems.

Theorem 1.4.1 (Algebraic Form of Logic Equations and their Solutions)
It is always possible to convert a logic equation to an equivalent algebraic form:

$$f(x_1, \ldots, x_n) = b, \tag{1.4.16}$$

where $f(x_1, \ldots, x_n)$ is a logic function as in Def. 1.1.2 and $b \in \mathcal{L}^2$ is a fixed logic vector. Then, $f(x_1, \ldots, x_n)$ can be expressed as:

$$f(x_1, \ldots, x_n) = M_f \ltimes \mathbf{x} = b, \tag{1.4.17}$$

where $M_f \in \mathcal{L}^{2\times 2^n}$ is the structure matrix of $f(\cdot)$ and $\mathbf{x} = \ltimes_{i=1}^n x_i \in \mathcal{L}^{2^n}$.
The solution of the equation is a set of n constant logic vectors $c_1, \ldots, c_n \in \mathcal{L}^2$. In fact, $(x_1, \ldots, x_n) = (c_1, \ldots, c_n)$ implies that $f(c_1, \ldots, c_n) = b$. It is possible to say that $\mathbf{x} = \mathbf{c} = \ltimes_{i=1}^n c_i \in \mathcal{L}^{2^n}$ is the solution of $f(\mathbf{x}) = b$. Looking at eqn. (1.4.17), one can say that all the solutions are $\mathbf{x} = \delta_{2^n}^i$, with $i \in \{i \,|\, \mathrm{Col}_i(M_f) = b\}$.

An example is proposed to clarify the last theorem.

Example 1.4.3 (Logic Equation in Algebraic Form and its Solutions)
Let us consider again the logic function of Ex. 1.4.1. It is possible to write a logic equation involving this function as follows:

$$f(x_1, x_2, x_3) = M_f \ltimes \mathbf{x} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \ltimes \mathbf{x} = b. \tag{1.4.18}$$

Following the result of Thm. 1.4.1, one gets:

- for $b = \delta_2^2$, the solutions are:

$$\begin{cases} \mathbf{x} = \mathbf{c}_1^0 = \delta_8^1 \\ \mathbf{x} = \mathbf{c}_2^0 = \delta_8^5 \\ \mathbf{x} = \mathbf{c}_3^0 = \delta_8^7, \end{cases} \tag{1.4.19}$$

  then:

$$\mathbf{x} = \mathbf{c}^0 = \left\{\mathbf{c}_1^0, \mathbf{c}_2^0, \mathbf{c}_3^0\right\}; \tag{1.4.20}$$

- for $b = \delta_2^1$, the solutions are:

$$\begin{cases} \mathbf{x} = \mathbf{c}_1^1 = \delta_8^2 \\ \mathbf{x} = \mathbf{c}_2^1 = \delta_8^3 \\ \mathbf{x} = \mathbf{c}_3^1 = \delta_8^4 \\ \mathbf{x} = \mathbf{c}_4^1 = \delta_8^6 \\ \mathbf{x} = \mathbf{c}_5^1 = \delta_8^8, \end{cases} \tag{1.4.21}$$

  then:

$$\mathbf{x} = \mathbf{c}^1 = \left\{\mathbf{c}_1^1, \mathbf{c}_2^1, \mathbf{c}_3^1, \mathbf{c}_4^1, \mathbf{c}_5^1\right\}. \tag{1.4.22}$$

Theorem 1.4.2 (Algebraic form of Logic Systems and their Solutions)
It is always possible to rewrite a logic system in an equivalent algebraic form:

$$\mathbf{f}(\mathbf{x}) = \mathbf{b}, \tag{1.4.23}$$

where:

$$\begin{aligned} \mathbf{f}(\cdot) : \mathcal{L}^{2^n} &\to \mathcal{L}^{2^m} \\ \mathbf{x} &\mapsto \mathbf{f}(\mathbf{x}), \end{aligned} \tag{1.4.24}$$

$\mathbf{x} \in \mathcal{L}^{2^n}$ and $\mathbf{b} \in \mathcal{L}^{2^m}$.
Given $c_1, \ldots, c_n \in \mathcal{L}^2$, the solution of the system is $\mathbf{x} = \mathbf{c} = \ltimes_{i=1}^{n} c_i \in \mathcal{L}^{2^n}$ if $\mathbf{f}(\mathbf{c}) = \mathbf{b}$.

An example for the solutions of a logic system is proposed.

Example 1.4.4 (Logic System in Algebraic Form and its Solutions)
Using the same system as in Ex. 1.4.2, one can get the system described by logic vectors and matrices:

$$\begin{cases} f_1(x_1, x_2) & = M_\vee \ltimes M_\neg \ltimes x_1 \ltimes x_2 = M_\vee \ltimes M_\neg \ltimes \mathbf{x_1} = b_1 \\ f_2(x_2, x_3) & = M_\wedge \ltimes x_2 \ltimes M_\neg \ltimes x_3 = \\ & = M_\wedge \ltimes W_{[2,2]} \ltimes M_\neg \ltimes W_{[2,2]} \ltimes x_2 \ltimes x_3 = \\ & = M_\wedge \ltimes W_{[2,2]} \ltimes M_\neg \ltimes W_{[2,2]} \ltimes \mathbf{x_2} = b_2. \end{cases} \tag{1.4.25}$$

Then:

$$\mathbf{f}(\mathbf{x}) = M_\vee \ltimes M_\neg \ltimes x_1 \ltimes x_2 \ltimes M_\wedge \ltimes W_{[2,2]} \ltimes M_\neg \ltimes W_{[2,2]} \ltimes x_2 \ltimes x_3 = \mathbf{b}. \tag{1.4.26}$$

The resulting truth table, obtained translating Tab. 1.4.1 through logic vectors, is shown in Tab. 1.4.2.
The solutions of $\mathbf{f}(\mathbf{x}) = \mathbf{b}$, for each possible value of $\mathbf{b}$, are:

- for $\mathbf{b} = \delta_4^4$:

$$\begin{cases} \mathbf{x} = \mathbf{c}_1^4 = \delta_8^3 \\ \mathbf{x} = \mathbf{c}_2^4 = \delta_8^4, \end{cases} \tag{1.4.27}$$

  then, the solution set is:

$$\mathbf{x} = \mathbf{c}^4 = \left\{\mathbf{c}_1^4, \mathbf{c}_2^4\right\}; \tag{1.4.28}$$

- for $\mathbf{b} = \delta_4^3$, there are no acceptable solutions:

$$\mathbf{x} = \mathbf{c}^3 = \{\emptyset\}; \tag{1.4.29}$$

- for $\mathbf{b} = \delta_4^2$:

$$\begin{cases} \mathbf{x} = \mathbf{c}_1^2 = \delta_8^1 \\ \mathbf{x} = \mathbf{c}_2^2 = \delta_8^5 \\ \mathbf{x} = \mathbf{c}_3^2 = \delta_8^7 \\ \mathbf{x} = \mathbf{c}_4^2 = \delta_8^8, \end{cases} \tag{1.4.30}$$

  then, the solution set is:

$$\mathbf{x} = \mathbf{c}^2 = \left\{\mathbf{c}_1^2, \mathbf{c}_2^2, \mathbf{c}_3^2, \mathbf{c}_4^2\right\}; \tag{1.4.31}$$

x_1	x_2	x_3	$\mathbf{x} = \ltimes_{i=1}^{3} x_i$	$f_1 = M_\vee \ltimes M_\neg \ltimes x_1 \ltimes x_2$	$f_2 = M_\wedge \ltimes x_2 \ltimes M_\neg \ltimes x_3$	$\mathbf{f} = f_1 \ltimes f_2$
δ_2^2	δ_2^2	δ_2^2	δ_8^8	δ_2^1	δ_2^2	δ_4^2
δ_2^2	δ_2^2	δ_2^1	δ_8^7	δ_2^1	δ_2^2	δ_4^2
δ_2^2	δ_2^1	δ_2^2	δ_8^6	δ_2^1	δ_2^1	δ_4^1
δ_2^2	δ_2^1	δ_2^1	δ_8^5	δ_2^1	δ_2^2	δ_4^2
δ_2^1	δ_2^2	δ_2^2	δ_8^4	δ_2^2	δ_2^2	δ_4^4
δ_2^1	δ_2^2	δ_2^1	δ_8^3	δ_2^2	δ_2^2	δ_4^4
δ_2^1	δ_2^1	δ_2^2	δ_8^2	δ_2^1	δ_2^1	δ_4^1
δ_2^1	δ_2^1	δ_2^1	δ_8^1	δ_2^1	δ_2^2	δ_4^2

Table 1.4.2: Truth Table of a Logic System in Algebraic Form

- for $\mathbf{b} = \delta_4^1$:

$$\begin{cases} \mathbf{x} = \mathbf{c}_1^1 = \delta_8^2 \\ \mathbf{x} = \mathbf{c}_2^1 = \delta_8^6, \end{cases} \tag{1.4.32}$$

then, the solution set is:

$$\mathbf{x} = \mathbf{c}^1 = \left\{ \mathbf{c}_1^1,\, \mathbf{c}_2^1 \right\}. \tag{1.4.33}$$

In the following theorem it is shown how to describe a logic system in algebraic form in a more compact and elegant way.

Theorem 1.4.3 (Algebraic Form of a Logic System and its Solutions)
The logic system of eqn. (1.4.23) can be rewritten as:

$$\mathbf{f}(\mathbf{x}) = L \ltimes \mathbf{x} = \mathbf{b}, \tag{1.4.34}$$

where $L \in \mathcal{L}^{2^m \times 2^n}$ is the structure matrix of the system.

Let us now provide an example to prove the previous statement.

Example 1.4.5 (Logic System in Algebraic Form)
From the truth table in Tab. 1.4.2 of Ex. 1.4.4, one can compute:

$$\begin{cases} L \ltimes \delta_8^1 = \mathrm{Col}_1(L) = \delta_4^2 \\ L \ltimes \delta_8^2 = \mathrm{Col}_2(L) = \delta_4^2 \\ L \ltimes \delta_8^3 = \mathrm{Col}_3(L) = \delta_4^1 \\ L \ltimes \delta_8^4 = \mathrm{Col}_4(L) = \delta_4^2 \\ L \ltimes \delta_8^5 = \mathrm{Col}_5(L) = \delta_4^4 \\ L \ltimes \delta_8^6 = \mathrm{Col}_6(L) = \delta_4^4 \\ L \ltimes \delta_8^7 = \mathrm{Col}_7(L) = \delta_4^1 \\ L \ltimes \delta_8^8 = \mathrm{Col}_8(L) = \delta_4^2. \end{cases} \tag{1.4.35}$$

Then, the structure matrix L is as follows:

$$L = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{1.4.36}$$

The next theorem offers an alternative procedure to compute L.

Theorem 1.4.4 (Structure Matrix of a Logic System)
The matrix L can be found from the single structure matrices of the equations in the system.

The procedure of the theorem is clarified by the following example.

Example 1.4.6 (Logic System in Algebraic Form)
The matrix L previously derived in Ex. 1.4.5 can be computed starting from eqn. (1.4.26) as follows:

$$\begin{aligned}
\mathbf{f}(x_1,\, x_2,\, x_3) &= M_\vee \ltimes M_\neg \ltimes x_1 \ltimes x_2 \ltimes M_\wedge \ltimes W_{[2,\,2]} \ltimes M_\neg \ltimes W_{[2,\,2]} \ltimes x_2 \ltimes x_3 = \\
&= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ltimes x_1 \ltimes x_2 \ltimes \\
&\ltimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes x_2 \ltimes x_3 = \\
&= \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ltimes x_1 \ltimes x_2 \ltimes \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \ltimes x_2 \ltimes x_3 = \\
&= M_{f_1} \ltimes x_1 \ltimes x_2 \ltimes M_{f_2} \ltimes x_2 \ltimes x_3 = \\
&= M_{f_1} \ltimes (I_4 \otimes M_{f_2}) \ltimes x_1 \ltimes x_2 \ltimes x_2 \ltimes x_3 = \\
&= M_{f_1} \ltimes (I_4 \otimes M_{f_2}) \ltimes x_1 \ltimes x_2^2 \ltimes x_3 = \\
&= M_{f_1} \ltimes (I_4 \otimes M_{f_2}) \ltimes x_1 \ltimes M_r \ltimes x_2 \ltimes x_3 = \\
&= M_{f_1} \ltimes (I_4 \otimes M_{f_2}) \ltimes (I_2 \otimes M_r) \ltimes x_1 \ltimes x_2 \ltimes x_3 = \\
&= \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ltimes \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \right) \ltimes \\
&\ltimes \left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \right) \ltimes x_1 \ltimes x_2 \ltimes x_3 = \\
&= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \ltimes x_1 \ltimes x_2 \ltimes x_3 = \\
&= L \ltimes \mathbf{x} = b_1 \ltimes b_2 = \mathbf{b},
\end{aligned} \tag{1.4.37}$$

where L is the same matrix as in eqn. (1.4.36).

As this example just showed, computing all L.S.T.P.s is a long process. Fortunately, there is an alternative way to compute L. It requires some preliminary results. First of all, two new operators, called front-maintaining and rear-maintaining operators, are introduced in the following definition.

Definition 1.4.3 (Front-maintaining and Rear-maintaining Operators)
The front-maintaining operator is defined as follows:

$$D_f^{p,\,q} := I_p \otimes \mathbf{1}_q^T = \\ = \underbrace{\begin{bmatrix} \underbrace{\delta_p^1 \ \cdots \ \delta_p^1}_{p} & \underbrace{\delta_p^2 \ \cdots \ \delta_p^2}_{p} & \cdots & \underbrace{\delta_p^q \ \cdots \ \delta_p^q}_{p} \end{bmatrix}}_{p\,q} \in \mathcal{L}^{p\times p\,q}, \tag{1.4.38}$$

Instead, the rear-maintaining operator is defined as follows:

$$D_r^{p,\,q} := \mathbf{1}_p^T \otimes I_q = \\ = \underbrace{\begin{bmatrix} \underbrace{\delta_q^1 \ \delta_q^2 \ \cdots \ \delta_q^q}_{q} & \cdots & \underbrace{\delta_q^1 \ \delta_q^2 \ \cdots \ \delta_q^q}_{q} \end{bmatrix}}_{p\,q} \in \mathcal{L}^{q\times p\,q}. \tag{1.4.39}$$

Consider now two logic vectors $X \in \mathcal{L}^p$ and $Y \in \mathcal{L}^q$, with $p,\, q \in \mathbb{N}^*$. Then, it is true that:

$$\begin{cases} D_f^{p,\,q} XY = X \\ D_r^{p,\,q} XY = Y. \end{cases} \tag{1.4.40}$$

The new operators allow to include in a logic expression a fictitious logic vector that does not alter the logic function.
Some examples are presented for clarity.

Example 1.4.7 (Front-maintaining Operator)
Let us consider $x_1,\, x_2,\, x_3 \in \mathcal{L}^2$, so that $\mathbf{x} = x_1 \ltimes x_2 \ltimes x_3$, and a function $f(x_1,\, x_2)$ as follows:

$$f(x_1,\, x_2) = M_f^{'} \ltimes x_1 \ltimes x_2, \tag{1.4.41}$$

with $M_f^{'} \in \mathcal{L}^{2\times 4}$. The objective is to write f in terms of $\mathbf{x}$ as:

$$f(\mathbf{x}) = M_f \ltimes \mathbf{x}. \tag{1.4.42}$$

Using the front-maintaining operator, one gets:

$$\begin{aligned} f(\mathbf{x}) &= M_f^{'} \ltimes x_1 \ltimes x_2 = \\ &= M_f^{'} \ltimes D_f^{4,\,2} \ltimes x_1 \ltimes x_2 \ltimes x_3 = \\ &= M_f \ltimes \mathbf{x}, \end{aligned} \tag{1.4.43}$$

where $M_f = M_f^{'} \ltimes D_f^{4,\,2} \in \mathcal{L}^{2\times 4}$ and

$$D_f^{4,\,2} = I_4 \otimes \mathbf{1}_2^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{1.4.44}$$

Example 1.4.8 (Rear-maintaining Operator)
Let us consider $x_1,\, x_2,\, x_3 \in \mathcal{L}^2$, so that $\mathbf{x} = x_1 \ltimes x_2 \ltimes x_3$, and a function $f(x_2,\, x_3)$ as follows:

$$f(x_2,\, x_3) = M_f^{'} \ltimes x_2 \ltimes x_3, \tag{1.4.45}$$

with $M_f^{'} \in \mathcal{L}^{2\times 4}$. The objective is to write f in terms of $\mathbf{x}$ as:

$$f(\mathbf{x}) = M_f \ltimes \mathbf{x}. \tag{1.4.46}$$

Using the rear-maintaining operator, one gets:

$$\begin{aligned} f(\mathbf{x}) &= M_f^{'} \ltimes x_2 \ltimes x_3 = \\ &= M_f^{'} \ltimes D_r^{2,\,4} \ltimes x_1 \ltimes x_2 \ltimes x_3 = \\ &= M_f \ltimes \mathbf{x}, \end{aligned} \tag{1.4.47}$$

where $M_f = M_f^{'} \ltimes D_r^{2,\,4} \in \mathcal{L}^{2\times 4}$ and

$$D_r^{2,\,4} = \mathbf{1}_2^T \otimes I_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.4.48}$$

The last necessary tool to compute L in a new way is described in the following definition.

Definition 1.4.4 (Khatri-Rao Matrix Product)
Let $A \in \mathbb{R}^{r_1\times c}$ and $B \in \mathbb{R}^{r_2\times c}$ be two matrices, with r_1, r_2 and $c \in \mathbb{N}^*$. The Khatri-Rao matrix product is denoted by $*$ and defined as follows:

$$C = A * B := \begin{bmatrix}\mathrm{Col}_1(A) \otimes \mathrm{Col}_1(B) & \cdots & \mathrm{Col}_c(A) \otimes \mathrm{Col}_c(B)\end{bmatrix}. \tag{1.4.49}$$

The product has these properties:

- commutativity with scalars:

$$(\alpha A) * (\beta B) = \alpha\beta\,(A * B), \quad \alpha,\, \beta \in \mathbb{R}; \tag{1.4.50}$$

- associativity:

$$(A * B) * (C * D) = A * (B * C) * D; \tag{1.4.51}$$

- distributivity w.r.t. addition:

$$(A + B) * (C + D) = (A * C) + (A * D) + (B * C) + (B * D), \tag{1.4.52}$$

  when A has the same dimensions as B and C has the same dimensions as D.

The resulting matrix C has dimensions $r_1\, r_2 \times c$. It is important to notice that C is defined only if the two matrices A and B have the same number of columns.

Because of the last remark, it might happen that, to obtain the algebraic form of a logic system, some equations do not contain all system variables. If this is the case, the m structure matrices M_j, with $j = 1, \ldots, m$, might not have the same number of columns or be related to different variables. Either way, Khatri-Rao product could not be used. A solution is to include all system variables in each equation, using the operators introduced in Def. 1.4.3, and then rearranging the variables with the swap matrices.

Proposition 1.4.1 (Algebraic Form of a Logic System)
Consider a logic system as follows:

$$\begin{cases} f_1(x_1,\,\dots,\,x_n) = M_{f_1} \ltimes \mathbf{x} = b_1 \\ f_2(x_1,\,\dots,\,x_n) = M_{f_2} \ltimes \mathbf{x} = b_2 \\ \vdots \\ f_m(x_1,\,\dots,\,x_n) = M_{f_m} \ltimes \mathbf{x} = b_m, \end{cases} \tag{1.4.53}$$

involving $m \in \mathbb{N}^*$ logic functions $f_j(\cdot) : \mathcal{L}^{2^n} \to \mathcal{L}^2$, with $j = 1,\,\dots,\,m$, $n \in \mathbb{N}^*$ boolean variables $x_i \in \mathcal{L}^2$, with $i = 1,\,\dots,\,n$, and m boolean constants $b_j \in \mathcal{L}^2$. Each logic function $f_j(\cdot)$ has an associated structure matrix $M_{f_j} \in \mathcal{L}^{2\times 2^n}$ and each equation depends on all the variables x_i, i.e. on $\mathbf{x} = \ltimes_{i=1}^{n} \in \mathcal{L}^{2^n}$. Then, the structure matrix of the system, L, can be found by means of the Khatri-Rao product as follows:

$$\begin{aligned} L &= M_{f_1} * M_{f_2} * \cdots * M_{f_m} = \\ &= \left[\mathrm{Col}_1(M_{f_1}) \ltimes \cdots \ltimes \mathrm{Col}_1(M_{f_m}) \quad \cdots \quad \mathrm{Col}_{2^n}(M_{f_1}) \ltimes \cdots \ltimes \mathrm{Col}_{2^n}(M_{f_m})\right] = \\ &= \left[\mathrm{Col}_1(M_{f_1}) \otimes \cdots \otimes \mathrm{Col}_1(M_{f_m}) \quad \cdots \quad \mathrm{Col}_{2^n}(M_{f_1}) \otimes \cdots \otimes \mathrm{Col}_{2^n}(M_{f_m})\right]. \end{aligned} \tag{1.4.54}$$

An example is now proposed.

Example 1.4.9 (L computation)
Using the logic system of Ex. 1.4.4, one can write:

$$\begin{cases} f_1 = M'_{f_1} \ltimes x_1 \ltimes x_2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ltimes x_1 \ltimes x_2 = M'_{f_1} \ltimes D_f^{4,\,2} \ltimes x_1 \ltimes x_2 \ltimes x_3 \\ f_2 = M'_{f_2} \ltimes x_2 \ltimes x_3 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \ltimes x_2 \ltimes x_3 = M'_{f_2} \ltimes D_r^{2,\,4} \ltimes x_1 \ltimes x_2 \ltimes x_3. \end{cases} \tag{1.4.55}$$

The structure matrices of each equation are:

$$\begin{cases} M_{f_1} &= M'_{f_1} \ltimes D_f^{4,\,2} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\ M_{f_2} &= M'_{f_2} \ltimes D_r^{2,\,4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \end{cases} \tag{1.4.56}$$

Then, the structure matrix of the system is:

$$L = M_{f_1} * M_{f_2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{1.4.57}$$

which coincides with the matrix in eqn. (1.4.36).

As for a logic equation, a method to find the solutions of a logic system as in eqn. (1.4.34) is presented in the following theorem.

Theorem 1.4.5 (Solution of a Logic System in Algebraic Form)
A logic system as in eqn. (1.4.34) has solutions if $\exists\, i \in \{1, \dots, m\}$ s.t. $\mathbf{b} = \mathrm{Col}_i(L)$. The solutions are $\mathbf{x} = \delta_{2^n}^i$, with $i \in \{i \,|\, \mathrm{Col}_i(L) = b\}$.

1.5 Non-negative Matrices

Logic vectors and matrices can only contain ones and zeros, by definition. This means that they are composed of non-negative elements. Considering the non-negative matrices $M = [m]_{i,j} \in \mathbb{R}^{r \times c}$, with $0 < i \leq r$, $0 < j \leq c$, and $r, c \in \mathbb{N}^*$, one can classify them in three categories:

1. non-negative: if $m_{i,j} \geq 0, \forall i, j$, M is non-negative and is denoted by $M \geq 0$;
2. positive: if M is non-negative and $\exists\, i, j$ s.t. $m_{i,j} > 0$, M is positive and is denoted by $M > 0$;
3. strictly positive: if $m_{i,j} > 0, \forall i, j$, M is strictly positive and is denoted by $M \gg 0$.

The same can be said about vectors.
Another classification for non-negative square matrices $M = [m]_{i,j} \in \mathbb{R}^{n \times n}$, with $n \in \mathbb{N}^*$ is the following:

- primitive: if $\exists\, s \in \mathbb{N}$ s.t. $M^s \gg 0$, then M is said to be a primitive matrix;
- irreducible: if $\forall i, j \in \{1, \dots, n\} \exists\, s_{i,j} \in \mathbb{N}$ s.t. $[M^{s_{i,j}}]_{i,j} > 0$, then M is irreducible. For a logic matrix $M \in \mathcal{L}^{n \times n}$, the irreducibility can be verified as follows:
$$\mathbb{M} := \bigvee_{s=0}^{n-1} M^s, \tag{1.5.1}$$
and M is irreducible if $\mathbb{M}$ has all entries equal to one.
- reducible: if M is not irreducible, then M is said to be reducible.

The references examined to write this chapter are the following: [10], [11], [12], [4], [13], [14], [15], [16].

Chapter 2

Boolean Networks

This chapter focuses on Boolean Networks (B.N.s), dynamical systems with logic state variables that evolve through logic functions. Moreover, these systems do not have (boolean) inputs unlike Boolean Control Networks (B.C.N.s), that will be treated in the next chapter. In this thesis, boolean networks are time-invariant, discrete-time and have a finite number of nodes. In this chapter we assume that their update mechanism is deterministic, while probabilistic boolean networks (P.B.N.s) will be discussed later on.

2.1 Boolean Networks Dynamics

The structure of a boolean network is defined in the following.

Definition 2.1.1 (Boolean Network)
A B.N. is a dynamical system involving boolean state variables that can influence each other by means of logic functions. The state variables are $\chi_1(t)$, $\chi_2(t)$, $\dots$, $\chi_n(t) \in \mathcal{B}$, with $n \in \mathbb{N}^*$. The evolution of the entire network is determined by a set of n logic first order difference equations:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_{1_1}(t),\, \chi_{2_1}(t),\, \dots,\, \chi_{n_1}(t)\right) \\ \chi_2(t+1) = f_2\left(\chi_{1_2}(t),\, \chi_{2_2}(t),\, \dots,\, \chi_{n_2}(t)\right) \\ \vdots \\ \chi_n(t+1) = f_n\left(\chi_{1_n}(t),\, \chi_{2_n}(t),\, \dots,\, \chi_{n_n}(t)\right), \end{cases} \tag{2.1.1}$$

where $f_i(\cdot) : \mathcal{B}^n \to \mathcal{B}$, $i \in \{1, n\}$ are the logic functions. It can happen that an equation does not depend on all logic variables, in fact, χ_{i_j} represents the i-th variable in the j-th equation, with i_j, $j \in \{1, n\}$ and $0 < n_j \leq n$.

An example of boolean network is presented next.

Example 2.1.1 (Boolean Network)
Let χ_1, χ_2 and $\chi_3 \in \mathcal{B}$ be the state variables of a B.N.. Then, a possible system of equations describing the B.N. can be as follows:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t)\right) = \chi_1(t) \vee \chi_2(t) \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_3(t)\right) = \chi_1(t) \wedge \neg\chi_3(t) \\ \chi_3(t+1) = f_3\left(\chi_2(t),\, \chi_3(t)\right) = \neg\chi_2(t) \wedge \chi_3(t). \end{cases} \tag{2.1.2}$$

Since a B.N. is a logic system, it can be converted to its algebraic form using the procedures explained in the previous chapter. This is formally expressed in the following proposition.

Proposition 2.1.1 (Boolean Networks in Algebraic Form)

$$\begin{cases} x_1(t+1) = f_1(x_{1_1}(t),\, x_{2_1}(t),\, \dots,\, x_{n_1}(t)) = M'_{f_1} \ltimes \mathbf{x}_1(t) = M_{f_1} \ltimes \mathbf{x}(t) \\ x_2(t+1) = f_2(x_{1_2}(t),\, x_{2_2}(t),\, \dots,\, x_{n_2}(t)) = M'_{f_2} \ltimes \mathbf{x}_2(t) = M_{f_2} \ltimes \mathbf{x}(t) \\ \vdots \\ x_n(t+1) = f_n(x_{1_n}(t),\, x_{2_n}(t),\, \dots,\, x_{n_n}(t)) = M'_{f_n} \ltimes \mathbf{x}_n(t) = M_{f_n} \ltimes \mathbf{x}(t), \end{cases} \tag{2.1.3}$$

where $x_{i_j}(\cdot) \in \mathcal{L}^2$, $f_j(\cdot) : \mathcal{L}^{2^{n_j}} \to \mathcal{L}^2$, $M'_{f_j} \in \mathcal{L}^{2\times 2^{n_j}}$, $M_{f_j} \in \mathcal{L}^{2\times 2^n}$, $i_j,\, j \in \{1,\, n\}$, $0 < n_j \le n$, $\mathbf{x}_j(t) = \ltimes_{i_j=1_j}^{n_j} x_{i_j}(t) \in \mathcal{L}^{2^{n_j}}$ and $\mathbf{x}(t) = \ltimes_{i=1}^{n} x_i(t) \in \mathcal{L}^{2^n}$.
To summarize this expression, one can rewrite the B.N. as:

$$\mathbf{x}(t+1) = L \ltimes \mathbf{x}(t), \tag{2.1.4}$$

where $L \in \mathcal{L}^{2^n \times 2^n}$ is the structure matrix.

An example is shown for clarity.

Example 2.1.2 (Boolean Networks in Algebraic Form)
Consider Ex. 2.1.1. Using Thm. 1.4.1, the B.N. becomes:

$$\begin{cases} x_1(t+1) &= M_\vee \ltimes x_1(t) \ltimes x_2(t) = M'_{f_1} \ltimes \mathbf{x}_1(t) \\ x_2(t+1) &= M_\wedge \ltimes x_1(t) \ltimes M_\neg \ltimes x_3(t) = \\ &= M_\wedge \ltimes (I_2 \otimes M_\neg) \ltimes \mathbf{x}_2(t) = M'_{f_2} \ltimes \mathbf{x}_2(t) \\ x_3(t+1) &= M_\wedge \ltimes M_\neg \ltimes x_2(t) \ltimes x_3(t) = M'_{f_3} \ltimes \mathbf{x}_3(t), \end{cases} \tag{2.1.5}$$

where:

$$\begin{cases} M'_{f_1} &= \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ M'_{f_2} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \ltimes \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \\ M'_{f_3} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}. \end{cases} \tag{2.1.6}$$

The next step is to add the missing state variables to each equation, using Def. 1.4.3:

$$\begin{cases} x_1(t+1) &= M'_{f_1} \ltimes \mathbf{x}_1(t) = M'_{f_1} \ltimes D_f^{4,\,2} \ltimes \mathbf{x}_1(t) \ltimes x_3(t) \\ &= M_{f_1} \ltimes \mathbf{x}(t) \\ x_2(t+1) &= M'_{f_2} \ltimes \mathbf{x}_2(t) = M'_{f_2} \ltimes D_f^{2,\,2} \ltimes x_1(t) \ltimes x_2(t) \ltimes x_3(t) \\ &= M_{f_2} \ltimes \mathbf{x}(t) \\ x_3(t+1) &= M'_{f_3} \ltimes \mathbf{x}_3(t) = M'_{f_3} \ltimes D_r^{2,\,4} \ltimes x_1(t) \ltimes \mathbf{x}_3(t) \\ &= M_{f_3} \ltimes \mathbf{x}(t), \end{cases} \tag{2.1.7}$$

where:

$$\begin{cases} M_{f_1} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \\ M_{f_2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \\ M_{f_3} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}. \end{cases} \tag{2.1.8}$$

Finally, one can compute the structure matrix of the network by means of the Khatri-Rao product:

$$\begin{aligned} L = M_{f_1} * M_{f_2} * M_{f_3} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \\ &= \delta_8 \begin{bmatrix} 4 & 2 & 3 & 2 & 4 & 4 & 7 & 8 \end{bmatrix}. \end{aligned} \tag{2.1.9}$$

Theorem 2.1.1 (Boolean Network Nodes Dynamics)
The algebraic form of eqn. (2.1.4) completely describes the dynamical behaviour of the $n \in \mathbb{N}^*$ variables of the B.N.. In fact:

$$\begin{aligned} \mathbf{x}(t) &= L \ltimes \mathbf{x}(t-1) = \\ &= L^2 \ltimes \mathbf{x}(t-2) = \\ &\vdots \\ &= L^{t-1} \ltimes \mathbf{x}(1) = \\ &= L^t \ltimes \mathbf{x}(0). \end{aligned} \tag{2.1.10}$$

From eqn. (2.1.3), it is true that $\forall\, i = 1, \ldots, n$:

$$x_i(t) = M_{f_i} \ltimes \mathbf{x}(t-1), \tag{2.1.11}$$

then:

$$x_i(t) = M_{f_i} \ltimes L \ltimes \mathbf{x}(t-2) = M_{f_i} \ltimes L^{t-1} \ltimes \mathbf{x}(0). \tag{2.1.12}$$

From the structure matrix L and the initial state condition $\mathbf{x}(0)$, it is possible to retrieve the evolution of each node of the B.N..

Since a B.N. described by eqn. (2.1.3) is time-invariant and deterministic, given an initial state condition $\mathbf{x}(0) \in \mathcal{L}^{2^n}$, the state trajectory $\mathbf{x}(t)$, with $t \in \mathbb{Z}$, is univocally determined. The structure matrix L, in fact, is unique and does not change over time. From an initial condition $\mathbf{x}(0)$, then, the state trajectory can be pre-determined using eqn. (2.1.10).

2.2 Graphical representation

Boolean Network nodes and their interactions are well represented by means of a directed graph. A brief presentation of a graph is given in the following.

Definition 2.2.1 (Graph)
A graph is a pair $\mathbf{G} \triangleq (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} \triangleq \{V_i : i = 1, \ldots, n\}$, with $n \in \mathbb{N}^*$, is the set of nodes (also called vertices) and $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$ is the set of edges. There exists a connection from node V_i to node V_j if the edge $e_{i,j}$ belongs to $\mathbf{E}$.

Definition 2.2.2 (Directed and Undirected Graphs)
A graph is called undirected if every edge is bidirectional, that is $\forall\,(V_i, V_j) \in \mathbf{E}$ also $(V_j, V_i) \in \mathbf{E}$, and the edge connecting V_i and V_j is denoted by $V_i \leftrightarrow V_j$.
If a graph is not undirected, it is directed, and the edge connecting V_i to V_j is denoted by $V_i \rightarrow V_j$.

Every boolean network can be described as a directed graph.

Definition 2.2.3 (Network Graph of a Boolean Network)
Given a B.N., we associate with it a graph whose set of nodes is the set of boolean variables: $\mathbf{V} = \{\chi_1, \ldots, \chi_n\}$. A pair (χ_i, χ_j) is an edge belonging to $\mathbf{E}$ if $\chi_i(t)$ is an argument of $f_j(\cdot)$ in eqn. (2.1.1), thus the node $\chi_i(t)$ affects $\chi_j(t+1)$, with $0 < i, j \leq n$. This directed graph is called network graph of the B.N.. Moreover, thanks to the equivalence between boolean variables and logic vectors, and consequently between the eqns. (2.1.1) and (2.1.3), the nodes of a network graph can either be $\chi_i \in \mathcal{B}$ or $x_i \in \mathcal{L}^2$, with $0 < i \leq n$ and $n \in \mathbb{N}^*$.

The next definition introduces the possible degrees of a generic node.

Definition 2.2.4 (In-degree and Out-degree of a Node)
Given a graph as in Def. 2.2.1 and a node $V_i \in \mathbf{V}$, the following parameters are defined:

- in-degree: it is the number of incoming edges of V_i;
- out-degree: it is the number of outgoing edges of V_i;
- total degree: it is the sum of the in-degree and out-degree of V_i.

The interaction between nodes, namely the presence of edges in a graph, is embedded in the incidence matrix. The definition and algorithm below show how to move from the graph to the incidence matrix and vice-versa.

Definition 2.2.5 (From Graph to Incidence Matrix)
Given a graph as in Def. 2.2.1, we define incidence matrix the boolean matrix $\mathcal{I}(\mathbf{G}) \in \mathcal{B}^{n\times n}$ whose entries are defined based on the following rule:

$$[\mathcal{I}(\mathbf{G})]_{i,j} = \begin{cases} 1, & \text{if } (V_i, V_j) \in \mathbf{E} \\ 0, & \text{otherwise} \end{cases}, \ \forall\, 0 < i, j \leq n. \tag{2.2.1}$$

Algorithm 2.2.1 (From Incidence Matrix to Graph)
Given an incidence matrix $\mathcal{I}(\mathbf{G}) \in \mathcal{B}^{n\times n}$, it is always possible to retrieve the corresponding graph.
Looking at the incidence matrix, one gets the edges connecting the $n \in \mathbb{N}^*$ nodes as follows:

$$[\mathcal{I}(\mathbf{G})]_{i,\,j} = \begin{cases} 1 \rightarrow (V_i,\, V_j) \in \mathbf{E} \\ 0 \rightarrow (V_i,\, V_j) \notin \mathbf{E} \end{cases} ,\ \forall\, 0 < i,\, j \leq n. \tag{2.2.2}$$

The next proposition shows an alternative way to derive the incidence matrix.

Proposition 2.2.1 (From Boolean Network to Incidence Matrix)
The incidence matrix $\mathcal{I}(\mathbf{G})$ can be directly derived from the logic equations of a B.N. as in eqn. (2.1.3) in the following way:

$$[\mathcal{I}(\mathbf{G})]_{i,\,j} = \begin{cases} 1, & \text{if } f_j(\cdot) \text{ depends on } x_i(t) \\ 0, & \text{otherwise} \end{cases} ,\ \forall\, 0 < i,\, j \leq n. \tag{2.2.3}$$

It is now clear that the incidence matrix has been introduced because it provides the same information as a network graph.
An example is presented in the following.

Example 2.2.1 (Network Graph and Incidence Matrix)
Consider again Ex. 2.1.1. It is possible to build the network graph $\mathbf{G} \triangleq (\mathbf{V},\, \mathbf{E})$ applying Def. 2.2.3. The set of nodes is $\mathbf{V} = \{\chi_1,\, \chi_2,\, \chi_3\}$ and the set of edges is $\mathbf{E} = \{e_{1,1},\, e_{2,1},\, e_{1,2},\, e_{3,2},\, e_{2,3},\, e_{3,3}\}$. The incidence matrix can be retrieved either from Def. 2.2.5 or from Prop. 2.2.1 as follows:

$$\mathcal{I}(\mathbf{G}) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{2.2.4}$$

The resulting graph $\mathbf{G} \triangleq (\mathbf{V},\, \mathbf{E})$ is reported in Fig. 2.2.1.

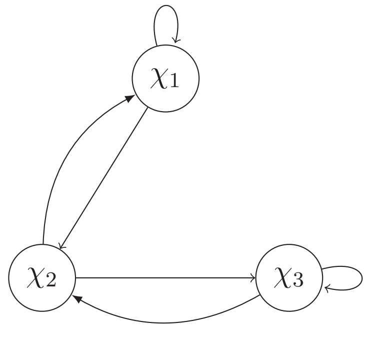

Figure 2.2.1: Network Graph of a Boolean Network

The correspondence between network graphs (or incidence matrices) and boolean networks is clarified in the following remark.

Remark 2.2.1 (Network Graphs)
A network graph and an incidence matrix do not represent univocally the B.N. of

eqn. (2.1.1). In fact, it is true that for a B.N. there exists a single network graph and a single incidence matrix. Instead, the formers can correspond to more than one B.N.. In two distinct B.N.s the dynamical behaviours can be different but the corresponding logic functions depend on the same nodes.
It is easy to see that network graphs and incidence matrices do not take into account the logic operators that connect the nodes of the B.N. but only their mutual influence.

Because network graphs and incidence matrices do not describe the network behaviour, a more powerful graph is now presented.

Definition 2.2.6 (State Graph of a Boolean Network)
Consider a B.N. in algebraic form, described as in eqn. (2.1.3). A directed graph $\mathcal{G} \triangleq \{\mathcal{V}, \mathcal{E}\}$ is the state graph of the B.N. if its vertices correspond to all the values that the state vector can assume, i.e. $\mathcal{V} = \{\delta_{2^n}^i : i = 1, \ldots, 2^n\}$, and the edges $\in \mathcal{E}$ are the elements $e_{i,j} = \left(\delta_{2^n}^i, \delta_{2^n}^j\right)$ s.t. $[L]_{j,i} = 1$, with $0 < i, j \leq 2^n$ (equivalently, $\mathrm{Col}_i(L) = \delta_{2^n}^j$).

Let us now explore a particular case of the previous definition.

Definition 2.2.7 (Self Loops of a State Graph)
Given the same configuration as in Def. 2.2.6, a particular edge exists when $[L]_{i,i} = 1$, with $0 < i \leq 2^n$ and $n \in \mathbb{N}^*$. The edge is called self-loop because the state of the B.N. remains unchanged over time:

$$\delta_{2^n}^i = \mathbf{x}(t+1) = L \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^n}^i. \tag{2.2.5}$$

In the state graph of a B.N. each node has exactly one outgoing edge which implies that its out-degree is equal to one. In fact, since L is deterministic and time-invariant, a state with value $\delta_{2^n}^i$ at time t evolves to another single state $\delta_{2^n}^j$ at time $t+1$, with $0 < i, j \leq 2^n$ and $n \in \mathbb{N}^*$. Algebraically, it holds that:

$$\delta_{2^n}^j = \mathbf{x}(t+1) = L \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^n}^i. \tag{2.2.6}$$

Instead, the in-degree of the nodes in a state graph has no restrictions. In fact, a node $\delta_{2^n}^j$ can be reached at time $t+1$ from more than one node at time t. For example, if $\mathbf{x}(t) = \delta_{2^n}^{i_1}$ or $\mathbf{x}(t) = \delta_{2^n}^{i_2}$, with $0 < i_1, i_2, j \leq 2^n$ and $i_1 \neq i_2$, then it follows that:

$$\delta_{2^n}^j = \mathbf{x}(t+1) = L \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^n}^{i_1} = L \ltimes \delta_{2^n}^{i_2}. \tag{2.2.7}$$

To conclude, since in a state graph there are 2^n nodes and each of them has one outgoing edge, there are exactly 2^n edges and the in-degree of each node is $\leq 2^n$.
The following definition introduces the idea of a path for a graph.

Definition 2.2.8 (Path in a Graph)
A path of the state graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, as in Def. 2.2.1, from node V_{i_1} to node $V_{i_{k+1}}$ is an ordered sequence of $k > 0$ distinct consecutive edges:

$$\left\{ \underbrace{(V_{i_1}, V_{i_2}), \ldots, (V_{i_k}, V_{i_{k+1}})}_{k} \right\}, \tag{2.2.8}$$

where the sequence of $k+1$ distinct vertices $V_{i_1}, \ldots, V_{i_{k+1}}$ is the vertex sequence of the path.

The path in a state graph is examined in the next definition.

Definition 2.2.9 (Path in a State Graph)
A path of the state graph $\mathcal{G} \triangleq \{\mathcal{V}, \mathcal{E}\}$, as in Def. 2.2.6, from node $\delta_{2^n}^{i_1}$ to node $\delta_{2^n}^{i_{k+1}}$ is an ordered sequence of $0 < k \leq 2^n$ distinct consecutive edges:

$$\left\{ \underbrace{\left(\delta_{2^n}^{i_1},\, \delta_{2^n}^{i_2}\right),\, \ldots,\, \left(\delta_{2^n}^{i_k},\, \delta_{2^n}^{i_{k+1}}\right)}_{k} \right\}, \tag{2.2.9}$$

where the sequence of $k+1$ distinct vertices $\delta_{2^n}^{i_1}, \ldots, \delta_{2^n}^{i_{k+1}}$ is the state sequence of the path.

Proposition 2.2.2 (Path in a State Graph)
Consider a B.N. as in eqn. (2.1.4) and two states $\delta_{2^n}^{i}$ and $\delta_{2^n}^{j} \in \mathcal{L}^{2^n}$. A directed path from $\delta_{2^n}^{i}$ to $\delta_{2^n}^{j}$ exists if and only if

$$\exists\, k,\ 0 < k \leq 2^n,\ n \in \mathbb{N}^* \,|\, \delta_{2^n}^{j} = L^k \ltimes \delta_{2^n}^{i}. \tag{2.2.10}$$

The following remark explains another relation between the structure matrix of a B.N. and its state graph.

Remark 2.2.2 (L and State Graph)
The structure matrix and the state graph of a B.N. as the one in eqn. (2.1.4) bring the same information about the network.

The same relation applies to a B.N. and its state graph.

Proposition 2.2.3 (Boolean Networks and State Graph)
Considering again the B.N. in eqn. (2.1.4), it is possible to say that there exists a one-to-one correspondence between the B.N. and its state graph.

The following example shows how to build the state graph of a boolean network.

Example 2.2.2 (State Graph)
Consider the B.N. of Ex. 2.1.2 which is fully characterized by the structure matrix L of eqn. (2.1.9). The set of vertices $\mathcal{V}$ and of edges $\mathcal{E}$ can be found looking at L. It follows that:

$$\begin{cases} \mathcal{V} = \left\{\delta_{2^3}^{i} : i = 1, 2, 3, 4, 5, 6, 7, 8\right\} \\ \mathcal{E} = \{\left(\delta_8^1, \delta_8^4\right), \left(\delta_8^2, \delta_8^2\right), \left(\delta_8^3, \delta_8^3\right), \left(\delta_8^4, \delta_8^2\right), \\ \qquad \left(\delta_8^5, \delta_8^4\right), \left(\delta_8^6, \delta_8^4\right), \left(\delta_8^7, \delta_8^7\right), \left(\delta_8^8, \delta_8^8\right)\}. \end{cases} \tag{2.2.11}$$

Then, the state graph is reported in Fig. 2.2.2.

2.3 Fixed Points and Limit Cycles

Fixed points and limit cycles are elements of fundamental importance because they are responsible for the asymptotic behaviour of a boolean network. Furthermore, they are crucial in the study of stability of a B.N.
The concepts of fixed point and limit cycle are now introduced.

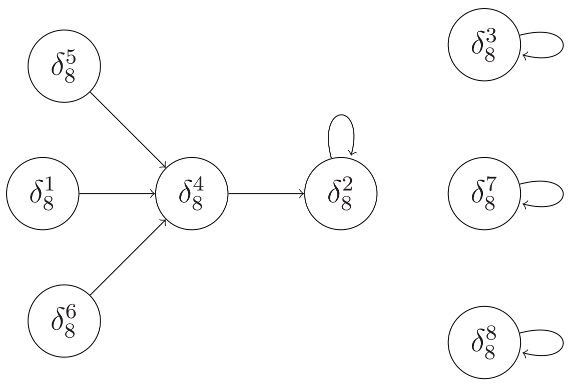

Figure 2.2.2: State Graph of a Boolean Network

Definition 2.3.1 (Fixed Point)
Given a B.N. described as in eqn. (2.1.4) and a time instant $t \in \mathbb{Z}$, a state $\delta_{2^n}^p$ is said to be a fixed point (or equilibrium point) of the B.N. if it holds that:

$$\delta_{2^n}^p = \mathbf{x}(t+1) = L \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^n}^p. \tag{2.3.1}$$

Definition 2.3.2 (Limit Cycle)
Given a B.N. described as in eqn. (2.1.4), a limit cycle of length $t \in \mathbb{Z}$ is an ordered sequence of t distinct states $\{\delta_{2^n}^c,\ L \ltimes \delta_{2^n}^c,\ \dots,\ L^{t-1} \ltimes \delta_{2^n}^c\}$ if it holds that:

$$\delta_{2^n}^c = \mathbf{x}(t) = L^t \ltimes \mathbf{x}(0) = L^t \ltimes \delta_{2^n}^c. \tag{2.3.2}$$

From these definitions it follows that a fixed point is a self-loop in the state graph of Def. 2.2.6, while a limit cycle is a loop in the state graph.
Let us now define the attractors which will be useful in the following.

Definition 2.3.3 (Attractors)
In a B.N. as in eqn. (2.1.4), an attractor is either a fixed point or a limit cycle. Therefore, there can be multiple attractors in this B.N..

The following proposition holds true because the B.N.s of this chapter are deterministic and time-invariant.

Proposition 2.3.1 (Convergence of a Boolean Network)
Given a deterministic and time-invariant B.N. as in eqn. (2.1.4), its dynamics converge to a specific attractor as in Def. 2.3.3. In detail, an initial state condition $\mathbf{x}(0) = \delta_{2^n}^i$, with $0 < i \leq 2^n$, converges to a fixed point or to a limit cycle. The states which are not fixed points or part of limit cycles are called transient states.

As a further consideration, a transient state converges to a single attractor and an attractive state belongs to one and only one limit cycle.
Thanks to the structure matrix L it is possible to identify the states that are fixed points and their number.

Theorem 2.3.1 (Fixed Points of a Boolean Network)
Given a B.N as in eqn. (2.1.4), the state $\delta_{2^n}^i = \delta_{2^n}^p \in \mathcal{L}^{2^n}$ is a fixed point if the i-th element of the diagonal of the structure matrix is unitary, namely $[L]_{i,i} = 1$. Then, the number N_p of fixed points can be found as follows:

$$N_p := \text{Trace}(L). \tag{2.3.3}$$

A fixed point can be considered as a limit cycle of length one. The identification of limit cycles of length greater than one is harder and it requires a preliminary concept.

Definition 2.3.4 (Proper Factors)
Given two numbers a and $b \in \mathbb{N}^*$, b is said to be a proper factor of a if $\frac{a}{b} \in \mathbb{N}^*$ and $b < a$. The set of all proper factors of a is denoted by $\mathcal{P}(a)$.

Example 2.3.1 (Proper Factors)
The proper factors of the number $a = 16$ are:

$$\mathcal{P}(16) = \{1,\, 2,\, 4,\, 8\}\,. \tag{2.3.4}$$

The next theorem shows an algorithm to find the number of limit cycles of a B.N..

Theorem 2.3.2 (Limit Cycles of a Boolean Network)
Consider a B.N. as in eqn. (2.1.4). The number N_s of limit cycles of length $s \in \mathbb{N}^*$ is:

$$N_s := \frac{\text{Trace}(L^s) - \sum_{k \in \mathcal{P}(s)} k\, N_k}{s}, \quad 2 \le s \le 2^n, \tag{2.3.5}$$

with $\mathcal{P}(s)$ the set of proper factors of s. When $s = 1$, the formula reduces to $N_1 = \text{Trace}(L) = N_p$ which is eqn. (2.3.3) for the number of fixed points.

Let us now explore the two limit cases for the limit cycles. In a B.N. with $n \in \mathbb{N}^*$ boolean variables the number of states is 2^n. Then, it might happen that a B.N. has a limit cycle of length 2^n, as shown in the following.
Consider the structure matrix $L = \delta_8 \begin{bmatrix} 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 \end{bmatrix}$. The state graph of the associated B.N. with $n = 3$ variables is reported in Fig. 2.3.1.

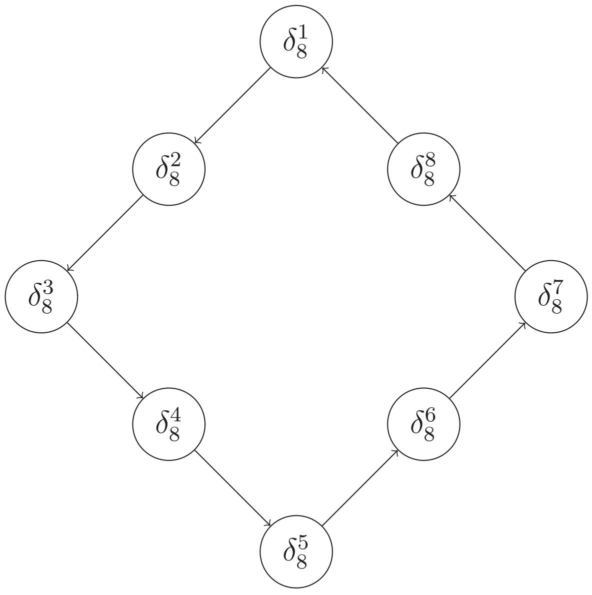

Figure 2.3.1: Limit Cycle of length 2^n of a Boolean Network

Let us now consider the opposite case: 2^n limit cycles of unitary length, or equivalently 2^n fixed points.
Consider the structure matrix $L = \delta_8 \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$. The state graph of the associated B.N. with $n = 3$ variables is reported in Fig. 2.3.2.
The following example explains how to compute the number of limit cycles in a B.N.. A brief definition of the powers of the structure matrix L is given before.

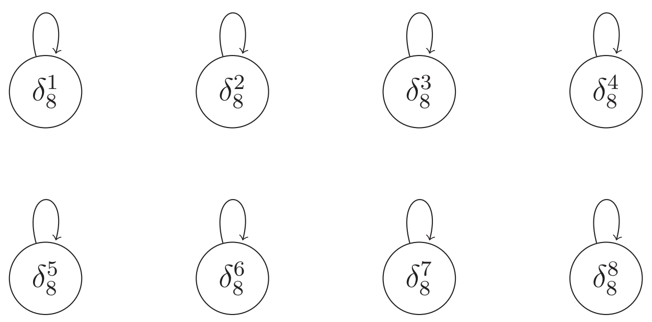

Figure 2.3.2: 2^n Limit Cycles of unitary length of a Boolean Network

Definition 2.3.5 (Powers of L)
Given a B.N. as in eqn. (2.1.4), the $s \in \mathbb{N}^*$ power of the structure matrix L is found as follows:

$$\begin{aligned} L^s &= L^{s-1} \ltimes L = \\ &= L^{s-2} \ltimes L \ltimes L = \\ &\vdots \\ &= \underbrace{L \ltimes L \ltimes \ \dots \ \ltimes L}_{s}. \end{aligned} \tag{2.3.6}$$

Example 2.3.2 (Fixed Points and Limit Cycles)
Given a B.N. as in eqn. (2.1.4) with:

$$L = \delta_8 \begin{bmatrix} 5 & 2 & 7 & 1 & 8 & 2 & 3 & 4 \end{bmatrix}, \tag{2.3.7}$$

in order to find the number of limit cycles and their lengths, let us begin by computing the powers of L defined in Def. 2.3.5. Since the number of states is $2^3 = 8$, one needs to compute the first eight powers of L and then apply eqn. (2.3.5):

$$\begin{aligned}
L^1 &= \delta_8 \begin{bmatrix} 5 & 2 & 7 & 1 & 8 & 2 & 3 & 4 \end{bmatrix} \rightarrow N_1 = \frac{\text{Trace}(L^1)}{1} = 1; \\
L^2 &= \delta_8 \begin{bmatrix} 8 & 2 & 3 & 5 & 4 & 2 & 7 & 1 \end{bmatrix} \rightarrow N_2 = \frac{\text{Trace}(L^2) - 1\,N_1}{2} = 1; \\
L^3 &= \delta_8 \begin{bmatrix} 4 & 2 & 7 & 8 & 1 & 2 & 3 & 5 \end{bmatrix} \rightarrow N_3 = \frac{\text{Trace}(L^3) - 1\,N_1}{3} = 0; \\
L^4 &= \delta_8 \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 2 & 7 & 8 \end{bmatrix} \rightarrow N_4 = \frac{\text{Trace}(L^4) - 1\,N_1 - 2\,N_2}{4} = 1; \\
L^5 &= \delta_8 \begin{bmatrix} 5 & 2 & 7 & 1 & 8 & 2 & 3 & 4 \end{bmatrix} \rightarrow N_5 = \frac{\text{Trace}(L^5) - 1\,N_1}{5} = 0; \\
L^6 &= \delta_8 \begin{bmatrix} 8 & 2 & 3 & 5 & 4 & 2 & 7 & 1 \end{bmatrix} \rightarrow N_6 = \frac{\text{Trace}(L^6) - 1\,N_1 - 2\,N_2 - 3\,N_3}{6} = 0; \\
L^7 &= \delta_8 \begin{bmatrix} 4 & 2 & 7 & 8 & 1 & 2 & 3 & 5 \end{bmatrix} \rightarrow N_7 = \frac{\text{Trace}(L^7) - 1\,N_1}{7} = 0; \\
L^8 &= \delta_8 \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 2 & 7 & 8 \end{bmatrix} \rightarrow N_8 = \frac{\text{Trace}(L^8) - 1\,N_1 - 2\,N_2 - 4\,N_4}{8} = 0.
\end{aligned} \tag{2.3.8}$$

In this B.N., then, there are three limit cycles of lengths one, two and four, as confirmed by the state graph in Fig. 2.3.3. Since a state can belong to a single limit cycle, the computations for cycles of length equal to five or greater could have been avoided. In fact, the first three cycles include seven states. As a consequence, it is not possible to build a limit cycle of length equal to or greater than five.

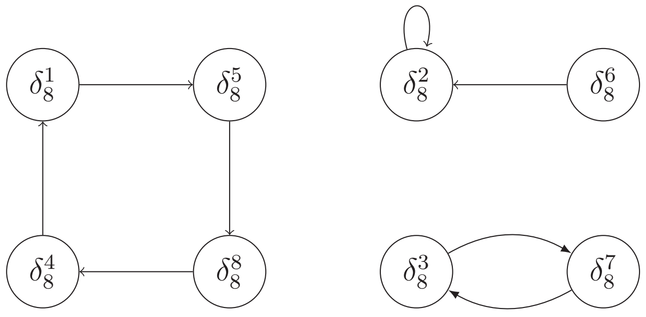

Figure 2.3.3: Limit Cycles and Fixed Points of a Boolean Network

It would be interesting now to define a method to find which nodes are associated with a specific limit cycle. This is presented in the following proposition.

Proposition 2.3.2 (Elements of a Limit Cycle)
Given a B.N. as in eqn. (2.1.4), let us define:

$$C_s = \left\{ i : [L^s]_{i,i} \right\}, \quad s = 1, \dots, 2^n, \tag{2.3.9}$$

and

$$D_s = C_s \cap \left(\bigcap_{i \in \mathcal{P}(s)} C_i^c \right), \tag{2.3.10}$$

where $C_i^c = U \setminus C_i$ is the complementary set of C_i w.r.t. the set $U = \{1, 2, \dots, 2^n\}$, with $n \in \mathbb{N}^*$. D_s is the set containing all elements that belong to a cycle of length s. This is equal to saying that $\{\delta_{2^n}^i, L \ltimes \delta_{2^n}^i, \dots, L^s \ltimes \delta_{2^n}^i\}$ is a limit cycle of length s if $i \in D_s$.

An example of the procedure is now shown.

Example 2.3.3 (Elements of a Limit Cycle)
Consider the previous Ex. 2.3.2. Let us compute the sets C_s and D_s as follows:

$$\begin{aligned} &C_1 = \{2\} \to D_1 = C_1 = \{2\} \\ &C_2 = \{2, 3, 7\} \to D_2 = C_2 \cap C_1^c = \{3, 7\} \\ &C_4 = \{1, 2, 3, 4, 5, 7, 8\} \to D_4 = C_4 \cap C_1^c \cap C_2^c = \{1, 4, 5, 8\}, \end{aligned} \tag{2.3.11}$$

with

$$\begin{aligned} U &= \{1, 2, 3, 4, 5, 6, 7, 8\} \\ C_1^c &= U \setminus C_1 = \{1, 3, 4, 5, 6, 7, 8\} \\ C_2^c &= U \setminus C_2 = \{1, 4, 5, 6, 8\}. \end{aligned} \tag{2.3.12}$$

The indices of the distinct sets D_s are the indices of the states that belong to a specific limit cycle, in the correct sequence. In this example, the limit cycles are:

$$\begin{aligned} &D_1 : \delta_8^2 \xrightarrow{L} \delta_8^2 \\ &D_2 : \delta_8^3 \xrightarrow{L} \delta_8^7 \\ &D_4 : \delta_8^1 \xrightarrow{L} \delta_8^5 \xrightarrow{L} \delta_8^8 \xrightarrow{L} \delta_8^4. \end{aligned} \tag{2.3.13}$$

The node δ_8^6 is a transient state and therefore does not belong to any limit cycle. Fig. 2.3.3 highlights the distinct limit cycles and the transient state.

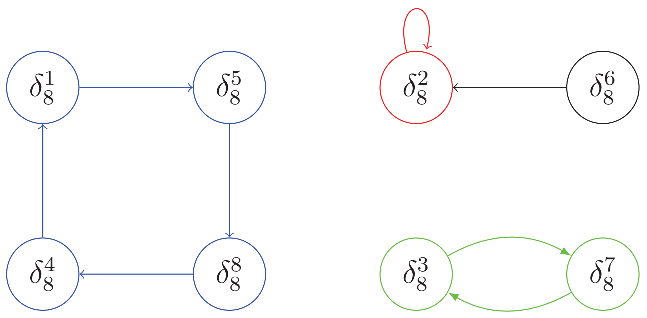

Figure 2.3.4: Limit Cycles and Transient State of a Boolean Network

2.4 Temporal analysis

Once transient states and the states belonging to attractors have been identified, the next step to fully understand a B.N is to find towards which attractor each transient state converges and also the number of steps to reach it. More generally, it is interesting to predict the convergence of the entire boolean network from each possible initial condition $\mathbf{x}(0) = \delta_{2^n}^i$, $i = 1, \ldots, 2^n$ with $n \in \mathbb{N}^*$.

Proposition 2.4.1 (Attractors Collection)
Let us denote by $m \in \mathbb{N}^*$ the number of limit cycles in a B.N described as in eqn. (2.1.4). Then denote by $\mathcal{C}_i$, with $i = 1, \ldots, m$, the set of nodes that belong to the i-th cycle. Finally let $\Omega = \bigcup_{i=1}^m \mathcal{C}_i$ be the set containing all the states that are part of a limit cycle. The sets $\mathcal{C}_i$ are disjoint, i.e. $\mathcal{C}_1 \cap \mathcal{C}_2 \cap \cdots \cap \mathcal{C}_m = \emptyset$. This holds because the states in a state graph have unitary out-degree.

Let us see an example.

Example 2.4.1 (Attractors Collection)
By referring to Ex. 2.3.3, one can obtain the following sets:

$$\begin{aligned} &D_1 \to \mathcal{C}_1 = \left\{\delta_8^2\right\} \\ &D_2 \to \mathcal{C}_2 = \left\{\delta_8^3,\, \delta_8^7\right\} \\ &D_4 \to \mathcal{C}_4 = \left\{\delta_8^1,\, \delta_8^4,\, \delta_8^5,\, \delta_8^8\right\}, \end{aligned} \tag{2.4.1}$$

which are disjoint sets, as expected. Then, Ω is as follows:

$$\Omega = \mathcal{C}_1 \cap \mathcal{C}_2 \cap \mathcal{C}_4 = \left\{\delta_8^1,\, \delta_8^2,\, \delta_8^3,\, \delta_8^4,\, \delta_8^5,\, \delta_8^7,\, \delta_8^8\right\}. \tag{2.4.2}$$

The following proposition introduces a time concept related to attractors.

Proposition 2.4.2 (Absorption Time of a State)
Given a B.N. described as in eqn. (2.1.4), for each initial condition $\mathbf{x}(0) = \delta_{2^n}^i$, $i = 1, \ldots, 2^n$ with $n \in \mathbb{N}^*$, the evolution converges to one attractor in a finite number of steps. The minimum time needed to reach that attractor is $T_t\left[\mathbf{x}(0)\right] \in \mathbb{N}$ and it holds that $\mathbf{x}\left(T_t\left[\mathbf{x}(0)\right]\right) \in \Omega$. Finally, $T_t\left[\mathbf{x}(0)\right]$ is called absorption time of the initial condition $\mathbf{x}(0)$.

If the initial state $\delta_{2^n}^i$ is already part of an attractor, its absorption time is:

$$T_t\left[\delta_{2^n}^i\right] = 0. \tag{2.4.3}$$

The absorption times of the previous example are evaluated in the following.

Example 2.4.2 (Attractor and Absorption Time of a State)
Let us consider again Ex. 2.3.3. By using Prop. 2.4.2 one can write the evolution of the trajectory for the first 2^3 steps from each initial condition:

$$\begin{aligned}
\delta_8^1 &: \left\{\delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \right\} \to h = 1,\, k = 5\\
\delta_8^2 &: \left\{\delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \right\} \to h = 1,\, k = 2\\
\delta_8^3 &: \left\{\delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \right\} \to h = 1,\, k = 3\\
\delta_8^4 &: \left\{\delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \right\} \to h = 1,\, k = 5\\
\delta_8^5 &: \left\{\delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \right\} \to h = 1,\, k = 5\\
\delta_8^6 &: \left\{\delta_8^6,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \delta_8^2,\, \right\} \to h = 2,\, k = 3\\
\delta_8^7 &: \left\{\delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \delta_8^7,\, \delta_8^3,\, \right\} \to h = 1,\, k = 3\\
\delta_8^8 &: \left\{\delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \delta_8^8,\, \delta_8^4,\, \delta_8^1,\, \delta_8^5,\, \right\} \to h = 1,\, k = 5,
\end{aligned} \tag{2.4.4}$$

where h and k are the indices of the first two repeated elements in a chain. Moreover, the length of a limit cycle is exactly $k - h$ and $T_t\left[\delta_{2^n}^i\right] = h - 1$. As a consequence:

$$\begin{aligned}
&\delta_8^1 \in \Omega \to T_t\left[\delta_{2^n}^1\right] = 0\\
&\delta_8^2 \in \Omega \to T_t\left[\delta_{2^n}^2\right] = 0\\
&\delta_8^3 \in \Omega \to T_t\left[\delta_{2^n}^3\right] = 0\\
&\delta_8^4 \in \Omega \to T_t\left[\delta_{2^n}^4\right] = 0\\
&\delta_8^5 \in \Omega \to T_t\left[\delta_{2^n}^5\right] = 0\\
&\delta_8^6 \xrightarrow{L} \delta_8^2 \in \Omega \to T_t\left[\delta_{2^n}^6\right] = 1\\
&\delta_8^7 \in \Omega \to T_t\left[\delta_{2^n}^7\right] = 0\\
&\delta_8^8 \in \Omega \to T_t\left[\delta_{2^n}^8\right] = 0.
\end{aligned} \tag{2.4.5}$$

The same results can be obtained observing the state graph of Fig. 2.3.4.

Prop. 2.4.2 introduced the concept of absorption time of a state. The generalization to the entire B.N. is done in the following definition.

Definition 2.4.1 (Absorption Time of a Boolean Network)
Given a B.N. as in eqn. (2.1.4), the absorption time of the B.N. is denoted by T_t and defined as follows:

$$T_t = \max_{\mathbf{x}(0) \in \mathcal{L}^{2^n}} \left\{T_t\left[\mathbf{x}(0)\right]\right\}. \tag{2.4.6}$$

According to the previous definition, the evolution of a B.N. enters a limit cycle after T_t steps. From Prop. 2.4.2, one can conclude that $T_t\left[\mathbf{x}(0)\right] = 2^n - 1$ in the worst case (Fig. 2.4.1, with $\mathbf{x}(0) = \delta_8^1$). Then, from Def. 2.4.1, $T_t \le 2^n - 1$.

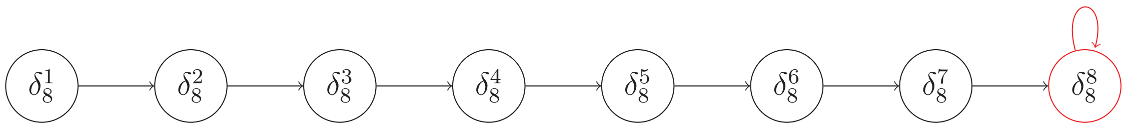
$$\begin{tikzcd}
\delta_8^1 \arrow[r] & \delta_8^2 \arrow[r] & \delta_8^3 \arrow[r] & \delta_8^4 \arrow[r] & \delta_8^5 \arrow[r] & \delta_8^6 \arrow[r] & \delta_8^7 \arrow[r] & \delta_8^8 \arrow[loop above]
\end{tikzcd}$$

Figure 2.4.1: Absorption Time, in the worst case, of a Boolean Network

Let us see an example.

Example 2.4.3 (Absorption Time of a Boolean Network)
Consider Ex. 2.4.2. Using eqn. (2.4.6), one easily gets:

$$T_t = 1. \tag{2.4.7}$$

In fact, the transient state δ_8^6 needs one step to enter the fixed point δ_8^2. In conclusion, the entire B.N. evolution belongs to a cycle after one step.

To further investigate the absorption time, a new concept is now introduced.

Proposition 2.4.3 (Powers of the Structure Matrix)
Let $r \triangleq 2^n \times 2^n$ denote the number of distinct values that the structure matrix L can take. This holds because there are 2^n columns and each can assume 2^n different values. Then, it is true that the sequence of the first $r+1$ powers of L

$$\{I_{2^n \times 2^n},\, L,\, \ldots,\, L^r\} \tag{2.4.8}$$

contains two elements which are equal.

Let us now define a new element.

Definition 2.4.2 (Smallest Exponent r_0)
Given a B.N. as in eqn. (2.1.4), r_0 is defined as the smallest exponent r such that:

$$\exists\, \tau,\, 0 < \tau < r : \, L^r = L^{r+\tau}. \tag{2.4.9}$$

An example is now shown.

Example 2.4.4 (Powers of the Structure Matrix and r_0)
Let us consider the structure matrix L of Ex. 2.4.3. According to Prop. 2.4.3, the first $r+1$ powers of L are related as follows:

$$\begin{aligned}
&L^0 = I_{8\times 8}\\
&L^1 = L^5 = L^9 \;\; = \cdots = L^{61}\\
&L^2 = L^6 = L^{10} = \cdots = L^{62}\\
&L^3 = L^7 = L^{11} = \cdots = L^{63}\\
&L^4 = L^8 = L^{12} = \cdots = L^{64}.
\end{aligned} \tag{2.4.10}$$

From Def. 2.4.2, one can obtain that $r_0 = 1$ and $\tau = 4$.

An insight about r_0 is presented in the following.

Proposition 2.4.4 (r_0)
Given a B.N. as in eqn. (2.1.4), for every initial condition $\mathbf{x}(0) \in \mathcal{L}^{2^n}$ the B.N. evolution becomes part of a limit cycle after (at most) r_0 steps. In fact, from Def. 2.4.2, $\exists\, \tau : L^{r_0} = L^{r_0+\tau}$. Then, the following holds true:

$$\mathbf{x}(r_0) = L^{r_0} \ltimes \mathbf{x}(0) = L^{r_0+\tau} \ltimes \mathbf{x}(0) = \mathbf{x}(r_0+\tau). \tag{2.4.11}$$

This shows that $\mathbf{x}(r_0)$ is part of a cycle, $\forall\, \mathbf{x}(0) \in \mathcal{L}^{2^n}$.

Remembering that all transient states belong to a specific cycle, r_0 must be in the interval $\{0, \ldots, 2^n - 1\}$.
The relation between the absorption time T_t and the exponent r_0 is explained in the following theorem.

Theorem 2.4.1 (Absorption Time and r_0)
Given a B.N. as in eqn. (2.1.4), the absorption time T_t and the exponent r_0 coincide, thus:

$$T_t = r_0. \tag{2.4.12}$$

Let us see an example for clarity.

Example 2.4.5 (Absorption Time and r_0)
From Exs. 2.4.3 and 2.4.4, it holds that:

$$T_t = r_0 = 1. \tag{2.4.13}$$

2.5 Boolean Network States Classification

This section focuses on the classification of the states of a B.N. so that new properties can be introduced.

2.5.1 Attractors and Basin of Attraction

Besides attractive and transient states, another classification is possible.

Definition 2.5.1 (Basin of Attraction)
A set $\mathcal{S}_i$ is the basin of attraction of $\mathcal{C}_i$ as defined in Prop. 2.4.1, if it contains the states of $\mathcal{C}_i$ and the states that converge to $\mathcal{C}_i$. A state $\mathbf{x}(0) = \delta_{2^n}^i$ belongs to $\mathcal{S}_i$ if:

$$\exists\, t : \mathbf{x}(t) = L^t \ltimes \mathbf{x}(0) = L^t \ltimes \delta_{2^n}^i \in \mathcal{C}_i, \quad \forall\, t \geq T_t\,[\mathbf{x}(0)]. \tag{2.5.1}$$

The following definition is needed in order to determine the basins of attraction of a B.N..

Definition 2.5.2 (Parent States in k-steps)
Consider two logic states $\delta_{2^n}^i$ and $\delta_{2^n}^j \in \mathcal{L}^{2^n}$. If it holds that:

$$\delta_{2^n}^j = L^k \ltimes \delta_{2^n}^i, \tag{2.5.2}$$

then $\delta_{2^n}^i$ is called a parent state in k steps of $\delta_{2^n}^j$, with $k \in \mathbb{N}^*$.
Since a state can be reached from many other states (the in-degree of a state is always $\leq 2^n$), the set of all parent states in k-steps of $\delta_{2^n}^j$ can be written as follows:

$$L^{-k}\left[\delta_{2^n}^j\right] = \left\{\delta_{2^n}^i \,|\, \delta_{2^n}^j = L^k \ltimes \delta_{2^n}^i\right\}. \tag{2.5.3}$$

To further generalize, given an attractor $\mathcal{C}$, the parent states in k-steps of $\mathcal{C}$ are:

$$L^{-k}\left[\mathcal{C}\right] = \left\{\delta_{2^n}^i \,|\, L^k \ltimes \delta_{2^n}^i \in \mathcal{C}\right\}. \tag{2.5.4}$$

In the case $k = 0$, the set of parent states in 0-steps is:

$$L^{-0}\left[\mathcal{C}\right] = \left\{\delta_{2^n}^i \,|\, L^0 \ltimes \delta_{2^n}^i = \delta_{2^n}^i \in \mathcal{C}\right\} = \mathcal{C}. \tag{2.5.5}$$

The following proposition is very important.

Proposition 2.5.1 (Basin of Attraction)
The basin of attraction $\mathcal{S}$ of an attractor $\mathcal{C}$ is as follows:

$$\mathcal{S} = \bigcup_{i=0}^{T_t} L^{-i}[\mathcal{C}], \tag{2.5.6}$$

and $\mathcal{C} \subseteq \mathcal{S}$. This is due to the fact that after T_t steps all nodes of a state graph belong to a limit cycle, specifically to $\mathcal{C}$.

Since the trajectory of a deterministic B.N. can be univocally predicted given any initial condition, each state is associated with one and only one attractor. Then, given an attractor C, the basin of attraction is also defined as follows:

$$\mathcal{S} = \left\{\delta_{2^n}^i \in \mathcal{L}^{2^n} \mid \text{if } \mathbf{x}(0) = \delta_{2^n}^i \text{ then } \mathbf{x}(t) \in \mathcal{C},\, \forall t \geq T_t\right\}. \tag{2.5.7}$$

The computation of $L^{-k}\left[\delta_{2^n}^j\right]$ is explained in the following proposition.

Proposition 2.5.2 (Parent States and Structure Matrices)
From Def. 2.5.2, the collection of parent states in k-steps of $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $k \in \mathbb{N}^*$, can be found as follows:

$$L^{-k}\left[\delta_{2^n}^j\right] = \left\{\delta_{2^n}^i \,|\, \mathrm{Col}_i(L^k) = \delta_{2^n}^j\right\}. \tag{2.5.8}$$

Let us consider an example of basins of attraction in a B.N..

Example 2.5.1 (Parent States and Structure Matrices)
Following Ex. 2.4.1, the basins of attractions are:

$$\begin{aligned}\mathcal{S}_1 &= \mathcal{C}_1 \cup \delta_8^6 = \left\{\delta_8^2,\, \delta_8^6\right\}\\ \mathcal{S}_2 &= \mathcal{C}_2 = \left\{\delta_8^3,\, \delta_8^7\right\}\\ \mathcal{S}_4 &= \mathcal{C}_4 = \left\{\delta_8^1,\, \delta_8^4,\, \delta_8^5,\, \delta_8^8\right\},\end{aligned} \tag{2.5.9}$$

which are all disjoint sets, constituting a complete partition of $\mathcal{L}^8$. In fact:

$$\mathcal{L}^8 = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4 = \left\{\delta_8^1,\, \delta_8^2,\, \delta_8^3,\, \delta_8^4,\, \delta_8^5,\, \delta_8^6,\, \delta_8^7,\, \delta_8^8\right\}. \tag{2.5.10}$$

The basins of attraction are highlighted in Fig. 2.5.1.

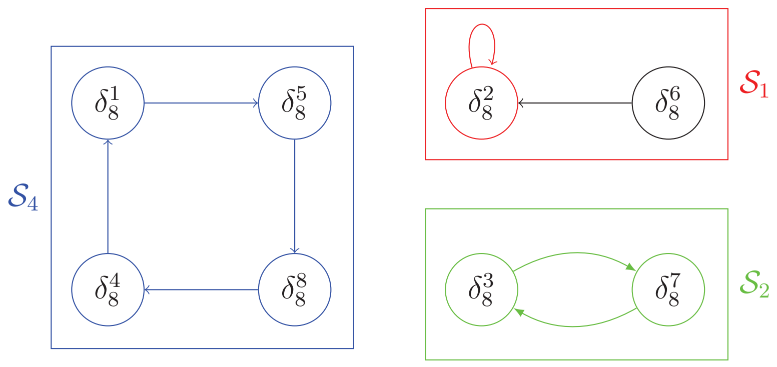

Figure 2.5.1: Basins of Attraction of a Boolean Network

2.5.2 Accessibility

Another classification for the states of a B.N. is called accessibility and it concerns the possibility that a state evolves into another one.

Definition 2.5.3 (Accessibility)
Consider a B.N. as in eqn. (2.1.4), a state $\delta_{2^n}^j \in \mathcal{L}^{2^n}$ is accessible from a state $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i,\, j \leq 2^n$, if $\exists\, k,\, 0 \leq k \leq 2^n - 1 \,|\, \delta_{2^n}^j = L^k \ltimes \delta_{2^n}^i$.

Accessibility is an equivalent concept to parenthood. In fact, $\delta_{2^n}^j$ is accessible from $\delta_{2^n}^i$ is another way of saying that $\delta_{2^n}^i$ is a parent of $\delta_{2^n}^j$ as in the following:

$$\exists\, k,\, 0 \leq k \leq 2^n - 1 \,|\, \delta_{2^n}^i \in L^{-k}\left[\delta_{2^n}^j\right]. \tag{2.5.11}$$

In the state graph of a B.N., the accessibility of $\delta_{2^n}^j$ from $\delta_{2^n}^i$ is equivalent to a directed path from $\delta_{2^n}^i$ to $\delta_{2^n}^j$.
It is important to notice that this concept is not symmetric. In fact, the accessibility of $\delta_{2^n}^j$ from $\delta_{2^n}^i$ does not imply the accessibility of $\delta_{2^n}^i$ from $\delta_{2^n}^j$.
A symmetric relation is given in the next definition.

Definition 2.5.4 (Communicability)
The communicability of two states $\delta_{2^n}^i$ and $\delta_{2^n}^j$ requires the accessibility in both directions. It must hold that:

$$\begin{cases} \exists\, k,\, 0 \leq k \leq 2^n - 1 \,|\, \delta_{2^n}^j = L^k \ltimes \delta_{2^n}^i \\ \exists\, h,\, 0 \leq h \leq 2^n - 1 \,|\, \delta_{2^n}^i = L^h \ltimes \delta_{2^n}^j. \end{cases} \tag{2.5.12}$$

This definition allows us to define the communication classes.

Definition 2.5.5 (Communication Classes)
A communication class $\mathcal{K}$ contains all the states that can communicate with each other. Specifically, a state $\delta_{2^n}^i$ belongs to the class $\mathcal{K}$ if it communicates with the other states in $\mathcal{K}$ and with no state outside $\mathcal{K}$.
Then, given two states $\delta_{2^n}^i$ and $\delta_{2^n}^j$, there are two possibilities:

- $\delta_{2^n}^i$ and $\delta_{2^n}^j$ belong to the same communication class because they communicate with each other;
- $\delta_{2^n}^i$ and $\delta_{2^n}^j$ belong to different communication classes because they do not communicate with each other. The following cases can happen:

- $\delta_{2^n}^j$ is accessible from $\delta_{2^n}^i$;
- $\delta_{2^n}^i$ is accessible from $\delta_{2^n}^j$
- $\delta_{2^n}^i$ and $\delta_{2^n}^j$ do not interact.

Let us now give a classification of communication classes.

Definition 2.5.6 (Types of Communication Classes)
A communication class $\mathcal{K}$ belongs to either one of the two following options:

- closed: when there are no transitions from nodes in $\mathcal{K}$ to nodes in other classes;
- transient: when there exists a path from a node in $\mathcal{K}$ to a node in another class.

The following proposition relates communication classes with limit cycles.

Proposition 2.5.3 (Communication Classes and Limit Cycles)
Given a B.N. as in eqn. (2.1.4), an attractor of the B.N. corresponds to a closed class in the state graph. This is why closed classes can also be called attractive classes.

Before showing an example about communication classes, let us remark that transient classes always contain only one state. In fact, a node of a transient class does not communicate with every other node and it accesses another class.

Example 2.5.2 (Communication Classes)
Consider again Ex. 2.3.2. The communication classes of this B.N. are:

$$\begin{aligned}
\mathcal{K}_1 &= \left\{\delta_8^1,\, \delta_8^4,\, \delta_8^5,\, \delta_8^8\right\} \\
\mathcal{K}_2 &= \left\{\delta_8^2\right\} \\
\mathcal{K}_3 &= \left\{\delta_8^3,\, \delta_8^7\right\} \\
\mathcal{K}_4 &= \left\{\delta_8^6\right\},
\end{aligned} \tag{2.5.13}$$

where $\mathcal{K}_1$ and $\mathcal{K}_3$ are closed classes of a limit cycle, $\mathcal{K}_2$ is a closed class of a fixed point and $\mathcal{K}_4$ is a transient class, that accesses $\mathcal{K}_2$. The communication classes are highlighted in Fig. 2.5.2, where closed classes are enclosed in colored rectangles.

Proposition 2.5.4 (Unique Limit Cycle)
In a B.N. there is a single communication class if and only if all the states in the B.N. belong to a limit cycle.

An example of the previous proposition is the state graph of Fig. 2.3.1.

2.6 Stability of a Boolean Network

The usefulness of rewriting the B.N. dynamics in algebraic form as in eqn. (2.1.4) is that it is possible to use the tools of Systems Theory to discuss the stability of the network.
The following two examples are important to recall the concept of equilibrium point which is crucial when studying the stability of a system.

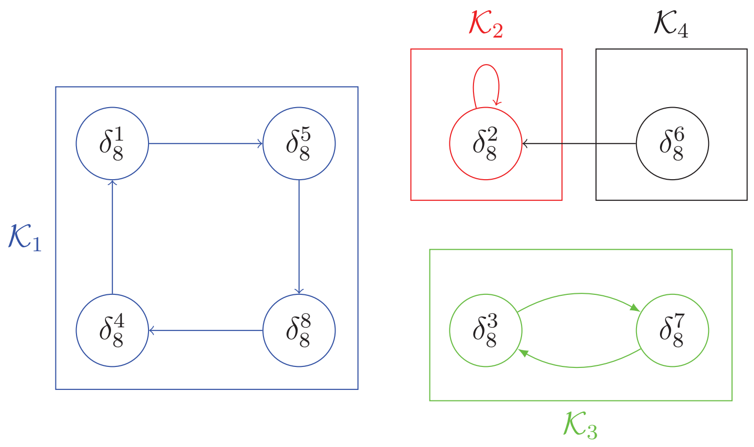

Figure 2.5.2: Communication Classes of a Boolean Network

Example 2.6.1 (Equilibrium Point)
Given a B.N. as in eqn. (2.1.4), with the structure matrix $L \in \mathcal{L}^{4\times 4}$ as follows:

$$L = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \tag{2.6.1}$$

In this case, looking at the trace of L, there are two equilibrium points: δ_4^2 and δ_4^4.

Example 2.6.2 (Equilibrium Point)
Given a B.N. as in eqn. (2.1.4), with the structure matrix $L \in \mathcal{L}^{4\times 4}$ as follows:

$$L = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.6.2}$$

In this case, the trace of L is null which means that there are no equilibrium points, but only limit cycles.

The stability of a B.N. is defined in the following definition.

Definition 2.6.1 (Global Stability)
A B.N. is said to be globally stable (G.S.) if the attractor set Ω contains only one fixed point. Equivalently, the B.N. is G.S. if there exists a fixed point $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, such that

$$\forall \mathbf{x}(0) \in \mathcal{L}^{2^n},\, \exists T_t \in \mathbb{Z}^+,\, T_t \geq 0 \,:\, \mathbf{x}(t) = \delta_{2^n}^i,\, \forall t \geq T_t. \tag{2.6.3}$$

In order to establish if a B.N. is globally stable, there is no need to find all attractors in the network. The next theorem exploits the structure matrix L to deduce the stability.

Theorem 2.6.1 (Global Stability)
A B.N. as in eqn. (2.1.4) is G.S. if and only if the following two equivalent statements hold:

1. The only power such that
$$\mathrm{Trace}\,[L^s] - \sum_{k\in\mathcal{P}(s)} kN_k > 0 \tag{2.6.4}$$
is $s = 1$, and it must be $\mathrm{Trace}[L] = 1$;
2. $\exists\, i,\, 0 < i \leq 2^n \,|\, L^k = \delta_{2^n}[i,\, \ldots,\, i],\ \forall\, k \geq T_t$, where $T_t \leq 2^n$ is the absorption time of the B.N..

An example is now shown for clarity.

Example 2.6.3 (Global Stability)
Consider a B.N. as in eqn. (2.1.4) with:
$$L = \delta_4 \begin{bmatrix} 2 & 2 & 4 & 1 \end{bmatrix}. \tag{2.6.5}$$

It is easy to see that the only element equal to one on the diagonal is the second one. Furthermore:
$$\begin{aligned} L^1 &= \delta_4 \begin{bmatrix} 2 & 2 & 4 & 1 \end{bmatrix} \rightarrow N_1 = \frac{\mathrm{Trace}\,(L^1)}{1} = 1 \\ L^2 &= \delta_4 \begin{bmatrix} 2 & 2 & 1 & 2 \end{bmatrix} \rightarrow N_2 = \frac{\mathrm{Trace}\,(L^2) - 1\,N_1}{2} = 0 \\ L^3 &= \delta_4 \begin{bmatrix} 2 & 2 & 2 & 2 \end{bmatrix} \rightarrow N_3 = \frac{\mathrm{Trace}\,(L^3) - 1\,N_1}{3} = 0 \\ L^4 &= \delta_4 \begin{bmatrix} 2 & 2 & 2 & 2 \end{bmatrix} \rightarrow N_4 = \frac{\mathrm{Trace}\,(L^4) - 1\,N_1 - 2\,N_2}{4} = 0. \end{aligned} \tag{2.6.6}$$

Applying Thm. 2.6.1, δ_4^2 results to be the only attractor, therefore proving the global stability of the B.N.. The state graph of this B.N. is reported in Fig. 2.6.1.

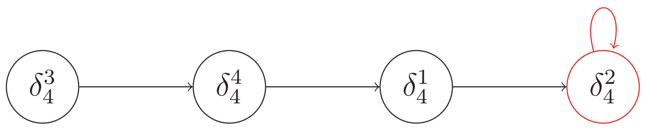
$$\begin{tikzcd} \delta_4^3 \arrow[r] & \delta_4^4 \arrow[r] & \delta_4^1 \arrow[r] & \delta_4^2 \arrow[loop above] \end{tikzcd}$$

Figure 2.6.1: Globally Stable Boolean Network

2.6.1 Globally Attractive Cycle

It is known that if a B.N. is not globally stable it must converge to one or more limit cycles. If for any initial condition the B.N. converges to a unique limit cycle (with a number of states greater than one), the latter is called globally attractive cycle (G.A.C.).

Definition 2.6.2 (Globally Attractive Cycle)
Given a B.N., a limit cycle $\mathcal{C}_G$ is globally attractive if the following holds:
$$\forall\, \mathbf{x}(0) \in \mathcal{L}^{2^n},\, \exists\, T \in \mathbb{Z} \,|\, \mathbf{x}(t) \in \mathcal{C}_G,\, \forall t \geq T. \tag{2.6.7}$$

The following proposition gives an insight of the globally attractive cycle.

Proposition 2.6.1 (Globally Attractive Cycle)
A cycle of a B.N. is globally attractive if and only if it is the unique cycle of the network. Furthermore, T is equal to the absorption time T_t.

Let us see an example in the following.

Example 2.6.4 (Globally Attractive Cycle)
Consider a B.N. as in eqn. (2.1.4) with:

$$L = \delta_8 \begin{bmatrix} 3 & 1 & 5 & 3 & 1 & 5 & 6 & 7 \end{bmatrix}. \tag{2.6.8}$$

From Def. 2.6.2 and Prop. 2.6.1, one can conclude that the limit cycle $\mathcal{C}_G = \{\delta_8^1,\, \delta_8^3,\, \delta_8^5\}$ is a G.A.C.. The state graph of this B.N. is reported in Fig. 2.6.2.

$$\begin{tikzcd}
\delta_8^2 \arrow[r] & \delta_8^1 \arrow[d] & \delta_8^5 \arrow[l] & \delta_8^6 \arrow[l] & \delta_8^7 \arrow[l] & \delta_8^8 \arrow[l] \\
\delta_8^4 \arrow[r] & \delta_8^3 \arrow[ru] & & & &
\end{tikzcd}$$

Figure 2.6.2: Globally Attractive Cycle in a Boolean Network

The references examined to write this chapter are the following: [4], [13], [14], [15], [17], [18].

Chapter 3

Boolean Control Networks

A B.N. is an autonomous system since no inputs act on the state variables. When adding boolean inputs a B.N. becomes a Boolean Control Network (B.C.N.) and it is now possible to control the evolution of the network by acting on these inputs.

3.1 Boolean Control Networks Dynamics

The structure of a boolean control network is defined in the following.

Definition 3.1.1 (Boolean Control Network)
A B.C.N. is a discrete time dynamical system involving $n \in \mathbb{N}^*$ boolean state variables $\chi_i(t) \in \mathcal{B}$ and $m \in \mathbb{N}^*$ boolean inputs $\mu_j \in \mathcal{B}$, with $0 < i \leq n$ and $0 < j \leq m$. The evolution of the network is determined by a set of n logic first order difference equations:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \ldots,\, \mu_m(t)\right) \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \ldots,\, \mu_m(t)\right) \\ \vdots \\ \chi_n(t+1) = f_n\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \ldots,\, \mu_m(t)\right), \end{cases} \tag{3.1.1}$$

where the logic functions $f_i(\cdot)$ are defined as follows:

$$\begin{gathered} f_i(\cdot) : \mathcal{B}^n \times \mathcal{B}^m \to \mathcal{B} \\ (\chi_1(t),\, \ldots,\, \chi_n(t)) \times (\mu_1(t),\, \ldots,\, \mu_m(t)) \mapsto f_i\left(\chi_1(t),\, \ldots,\, \chi_n(t);\, \mu_1(t),\, \ldots,\, \mu_m(t)\right). \end{gathered} \tag{3.1.2}$$

It is also possible to consider boolean outputs ψ_k as shown in the following:

$$\begin{cases} \psi_1(t) = h_1\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t)\right) \\ \vdots \\ \psi_p(t) = h_p\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t)\right), \end{cases} \tag{3.1.3}$$

where, $\forall\, k$ s.t. $0 < k \leq p$, the boolean functions $h_k(\cdot)$ are defined as follows:

$$\begin{gathered} h_k(\cdot) : \mathcal{B}^n \to \mathcal{B} \\ (\chi_1(t),\, \ldots,\, \chi_n(t)) \mapsto h_k\left(\chi_1(t),\, \ldots,\, \chi_n(t)\right). \end{gathered} \tag{3.1.4}$$

A B.C.N. can be rewritten in its algebraic form with the same method used for B.N.s. Then, one needs to transform all boolean state variables, inputs, outputs and functions into their algebraic representation. This is explained in the following proposition.

Proposition 3.1.1 (Boolean Control Network in Algebraic Form)
A B.C.N. as in eqns. (3.1.1) and (3.1.3) can be transformed into its equivalent algebraic form following Thm. 1.4.3:

$$\begin{cases} \mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ \mathbf{y}(t) = H \ltimes \mathbf{x}(t), \end{cases} \tag{3.1.5}$$

where $\mathbf{x}(\cdot) \in \mathcal{L}^{2^n}$ is the state vector, $\mathbf{u}(\cdot) \in \mathcal{L}^{2^m}$ is the input vector, $\mathbf{y}(\cdot) \in \mathcal{L}^{2^k}$ is the output vector, $L \in \mathcal{L}^{2^n \times 2^{n+m}}$ is the transition matrix of the B.C.N. and $H \in \mathcal{L}^{2^p \times 2^n}$. Using Thm. 1.4.2, one can obtain the following system:

$$\begin{cases} x_1(t+1) & = f_1(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) & = f_2(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ \vdots & \\ x_n(t+1) & = f_n(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_n} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \end{cases} \tag{3.1.6}$$

where $x_i(\cdot) \in \mathcal{L}^2$, $\forall\, i$, $0 < i \le n$, $u_j(\cdot) \in \mathcal{L}^2$, $\forall\, j$, $0 < j \le m$, $M_{f_i} \in \mathcal{L}^{2^n \times 2^{n+m}}$ and the logic functions $f_i(\cdot)$ are defined as follows:

$$\begin{gathered} f_i(\cdot) : \mathcal{L}^{2^n} \times \mathcal{L}^{2^m} \to \mathcal{L}^{2^n} \\ (x_1(t),\, \dots,\, x_n(t)) \times (u_1(t),\, \dots,\, u_m(t)) \mapsto f_i\left(x_1(t),\, \dots,\, x_n(t);\, u_1(t),\, \dots,\, u_m(t)\right). \end{gathered} \tag{3.1.7}$$

The same changes can be applied to the output equations as follows:

$$\begin{cases} y_1(t+1) & = h_1(x_1(t),\, x_2(t),\, \dots,\, x_n(t)) = M_{h_1} \ltimes \mathbf{x}(t) \\ y_2(t+1) & = h_2(x_1(t),\, x_2(t),\, \dots,\, x_n(t)) = M_{h_2} \ltimes \mathbf{x}(t) \\ \vdots & \\ y_p(t+1) & = h_p(x_1(t),\, x_2(t),\, \dots,\, x_n(t)) = M_{h_p} \ltimes \mathbf{x}(t), \end{cases} \tag{3.1.8}$$

where $y_k(\cdot) \in \mathcal{L}^2$, $\forall\, k$, $0 < k \le p$, $M_{h_k} \in \mathcal{L}^{2^p \times 2^n}$ and the logic functions $h_k(\cdot)$ are defined as follows:

$$\begin{gathered} h_k(\cdot) : \mathcal{L}^{2^n} \to \mathcal{L}^{2^p} \\ (x_1(t),\, \dots,\, x_n(t)) \mapsto h_k\left(x_1(t),\, \dots,\, x_n(t)\right). \end{gathered} \tag{3.1.9}$$

In eqn. (3.1.5), it is customary to put the input vector $\mathbf{u}(\cdot)$ before the state vector $\mathbf{x}(\cdot)$. Let us remark that this choice does not alter the dynamics of the B.C.N. since the two vectors can be exchanged by means of a proper swap matrix.

Example 3.1.1 (Boolean Control Network in Algebraic Form)
Given a B.C.N. with $n = 3$ state variables, $m = 2$ inputs and $p = 1$ outputs described as in the following:

$$\begin{cases} \chi_1(t+1) = (\chi_1(t) \wedge \chi_2(t)) \vee \mu_1(t) \\ \chi_2(t+1) = (\chi_2(t) \vee \chi_3(t)) \wedge \mu_2(t) \\ \chi_3(t+1) = \neg\chi_1(t) \wedge \chi_3(t) \\ \psi(t) = \chi_1(t) \vee \neg\chi_2(t), \end{cases} \tag{3.1.10}$$

it is possible to rewrite the system as:

$$\begin{cases} x_1(t+1) = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_3(t+1) = M_{f_3} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ y(t) = M_h \ltimes \mathbf{x}(t), \end{cases} \tag{3.1.11}$$

where $M_{f_i} \in \mathcal{L}^{2\times 32}$, with $1 \leq i \leq 3$ and $M_h \in \mathcal{L}^{2\times 8}$. Finally, the matrices L and H in the B.C.N. description (3.1.5) are:

$$\begin{cases} L = \delta_8\,[2\;\,2\;\,2\;\,4\;\,1\;\,2\;\,1\;\,4\;\,4\;\,4\;\,4\;\,4\;\,3\;\,4\;\,3\;\,4\;\,2\;\,2\;\,6\;\,8\;\,5\;\,6\;\,5\;\,8\;\,4\;\,4\;\,8\;\,8\;\,7\;\,8\;\,7\;\,8] \\ H = M_h = \delta_8\,[2\;\,2\;\,1\;\,1\;\,2\;\,2\;\,2\;\,2], \end{cases} \tag{3.1.12}$$

where $L \in \mathcal{L}^{8\times 32}$ and $H \in \mathcal{L}^{2\times 8}$.

3.1.1 Boolean Control Networks as Switched Systems

With the introduction of inputs, the state graph of a B.N. changes depending on the values of these inputs. Therefore, it is possible to interpret a B.C.N. as a boolean switched system that varies among 2^m different boolean networks.
Let us recall that switched systems are a class of hybrid systems which consist of a family of subsystems and some law regulating the switching among them, depending on various factors. In this case, the factor that makes the system switch is the value of the inputs, represented by the input vector $\mathbf{u}(t)$ at time $t \in \mathbb{Z}$. Algebraically, it holds:

$$\mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = L\,[\mathbf{u}(t)] \ltimes \mathbf{x}(t) = L_j \ltimes \mathbf{x}(t), \tag{3.1.13}$$

where $L_j \in \mathcal{L}^{2^n \times 2^n}$ is the structure matrix of the active B.N. at time t. L_j is selected by the input value $\mathbf{u}(t) = \delta_{2^m}^j$, with $0 < j \leq 2^m$. Then, L can be rewritten as:

$$L = \begin{bmatrix} L_1 & L_2 & \dots & L_{2^m} \end{bmatrix}. \tag{3.1.14}$$

Therefore, the B.C.N. can be written as a switched system:

$$\mathbf{x}(t+1) = L_{\sigma(t)} \ltimes \mathbf{x}(t), \tag{3.1.15}$$

where $\sigma(t)$ is the switching input taking values in $\{1, \dots, 2^m\}$.
Let us now consider an example.

Example 3.1.2 (Boolean Control Networks as Switched Systems)
Consider the same B.C.N. as in Ex. 3.1.1. Since there are $m = 2$ inputs in the network, i.e. $u_1(\cdot)$ and $u_2(\cdot)$, it means that there are $2^m = 4$ sub-networks represented by the four blocks in the matrix L:

$$L = \begin{bmatrix} L_1 & L_2 & L_3 & L_4 \end{bmatrix}, \tag{3.1.16}$$

where:

$$\begin{cases} L_1 = \begin{bmatrix} 2 & 2 & 2 & 4 & 1 & 2 & 1 & 4 \end{bmatrix} \\ L_2 = \begin{bmatrix} 4 & 4 & 4 & 4 & 3 & 4 & 3 & 4 \end{bmatrix} \\ L_3 = \begin{bmatrix} 2 & 2 & 6 & 8 & 5 & 6 & 5 & 8 \end{bmatrix} \\ L_4 = \begin{bmatrix} 4 & 4 & 8 & 8 & 7 & 8 & 7 & 8 \end{bmatrix}. \end{cases} \tag{3.1.17}$$

3.1.2 The Matrix L_{tot}

The transition matrix L, defined in Prop. 3.1.1, contains the information about the evolution of a B.C.N. for every possible value of the input vector $\mathbf{u}(\cdot)$, as previously clarified. The objective now is to adopt a smaller matrix w.r.t. L that brings the necessary information to study some properties about the evolution of the states. This matrix is called L_{tot} and is presented in the following definition.

Definition 3.1.2 (The matrix L_{tot})
Given a B.C.N. described as in eqn. (3.1.5), the matrix L_{tot} is defined as follows:

$$L_{tot} := \bigcup_{i=1}^{2^m} L_i \in \mathcal{L}^{2^n \times 2^n}. \tag{3.1.18}$$

Let us understand the meaning of this matrix. The $(j,\, i)$-th entry of L_{tot}, that is $[L_{tot}]_{j,\,i}$, is equal to one if and only if there exists an input $\mathbf{u}(t) = \delta_{2^m}^k \in \mathcal{L}^{2^m}$ such that the state $\mathbf{x}(t) = \delta_{2^n}^i \in \mathcal{L}^{2^n}$ evolves into $\mathbf{x}(t+1) = \delta_{2^n}^j \in \mathcal{L}^{2^n}$, with $0 < i,\, j \leq 2^n$, $0 < k \leq 2^m$ and $t \in \mathbb{Z}$. Algebraically, it must hold that:

$$\delta_{2^n}^j = \mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^m}^k \ltimes \mathbf{x}(t) = L_k \ltimes \mathbf{x}(t) = L_k \ltimes \delta_{2^n}^i. \tag{3.1.19}$$

This means that there exists at least one input such that $\delta_{2^n}^j$ is accessible from $\delta_{2^n}^i$ in one step. Then, the unitary entries in $\mathrm{Col}_i(L_{tot})$ represent the states $\delta_{2^n}^j$ that are reachable from $\delta_{2^n}^i$ in one step. As a consequence, the out-degree of the i-th node is:

$$\sum_{j=1}^{2^n} [L_{tot}]_{j,\,i} \leq 2^m. \tag{3.1.20}$$

For this reason, L_{tot} will prove to be useful to graphically represent the B.C.N..
Let us remark that L_{tot} is no longer a logic matrix but only a boolean one because its columns can be non-logic vectors. In fact, by construction of this matrix, the operations of sum and product are substituted with their logical equivalent $\vee$ and $\wedge$ respectively to preserve the boolean property of L_{tot} when the L.S.T.P. is used. An example of computation of L_{tot} is now shown.

Example 3.1.3 (The Matrix L_{tot})
By referring to the matrix L of eqn. (3.1.12), it is possible to compute L_{tot} using eqn. (3.1.18):

$$L_{tot} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}. \tag{3.1.21}$$

3.1.3 Graphical Representation

Also in the case of a B.C.N., it is possible to represent graphically the network by means of a state graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as in Def. 2.2.6. The set of nodes is $\mathcal{V} = \left\{\delta_{2^n}^i \in \mathcal{L}^{2^n},\, 0 < i \leq 2^n\right\}$ with $n \in \mathbb{N}^*$. The set of edges is slightly different as they are derived from the matrix L_{tot}. In fact, the oriented edge $e_{j,i} = (\delta_{2^n}^i, \delta_{2^n}^j)$ from the i-th node to the j-th node belongs to $\mathcal{E}$ if $[L_{tot}]_{j,i} = 1$.
Referring to eqn. (3.1.20), one can notice that in the state graph of a B.C.N. each node may have an out-degree greater than one due to the presence of inputs. Specifically, there can be at most 2^m outgoing edges for each node of the network because there are at most 2^m possible input values.
The next example clarifies these comments.

Example 3.1.4 (Graphical Representation)
Given a B.C.N. in algebraic form with two state variables and one input $\in \mathcal{L}^2$:

$$\mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \tag{3.1.22}$$

where $\mathbf{x}(\cdot) \in \mathcal{L}^4$, $\mathbf{u}(\cdot) \in \mathcal{L}^2$ and therefore $L \in \mathcal{L}^{4\times 8}$. The transition matrix L has the following entries:

$$L = \delta_4 \begin{bmatrix} 4 & 3 & 4 & 2 & 1 & 4 & 4 & 1 \end{bmatrix}. \tag{3.1.23}$$

If $\mathbf{u}(\cdot) = \delta_2^1$, the first four columns are selected, thus:

$$L_1 = \delta_4 \begin{bmatrix} 4 & 3 & 4 & 2 \end{bmatrix} \in \mathcal{L}^{4\times 2}. \tag{3.1.24}$$

Instead, if $\mathbf{u}(\cdot) = \delta_2^2$, the last four columns are selected:

$$L_2 = \delta_4 \begin{bmatrix} 1 & 4 & 4 & 1 \end{bmatrix} \in \mathcal{L}^{4\times 2}. \tag{3.1.25}$$

By using eqn. (3.1.18), it is possible to compute:

$$L_{tot} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}. \tag{3.1.26}$$

Thanks to the information embedded in L_{tot}, one can build the corresponding state graph $\mathcal{G}$:

$$\begin{cases} \mathcal{V} = \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\} \\ \mathcal{E} = \{(\delta_4^1,\, \delta_4^1),\, (\delta_4^1,\, \delta_4^4),\, (\delta_4^2,\, \delta_4^3),\, (\delta_4^2,\, \delta_4^4), \\ \qquad\quad (\delta_4^3,\, \delta_4^4),\, (\delta_4^4,\, \delta_4^1),\, (\delta_4^4,\, \delta_4^2)\}. \end{cases} \tag{3.1.27}$$

Finally, the state graph is reported in Fig. 3.1.1, where edges depending on distinct inputs are highlighted with different colors.

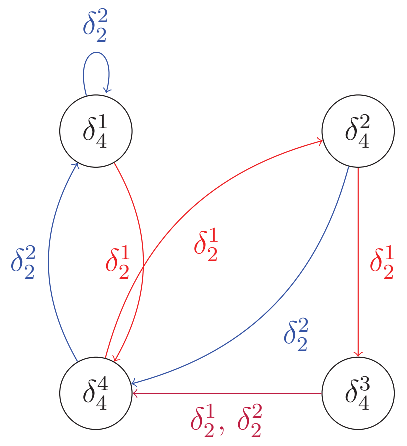

Figure 3.1.1: State Graph of a Boolean Control Network

3.1.4 Communication Classes

Since the inputs in a B.C.N. can change at every time instant $t \in \mathbb{Z}$, the convergence to an attractor is no more guaranteed. In fact, it might happen that, for a specific sequence of inputs, the evolution never converges to a fixed point or a limit cycle. Def. 2.5.5 of communication classes of B.N.s holds true also for B.C.N.s but the transient classes can contain more than one state. Anyway, as soon as the trajectory leaves a transient class it cannot re-enter it, otherwise it would be a closed class. In a B.N., once the trajectory enters a closed class, its evolution is periodic in the nodes belonging to that class, as stated in Prop. 2.5.3. In a B.C.N., this is no longer true because of the arbitrariness in the input choice.
Let us clarify this reasoning with an example.

Example 3.1.5 (Communication Classes of a Boolean Control Network)
Consider a B.C.N. as in eqn. (3.1.5) composed of two state variables and one input so that:

$$L = \delta_4 \begin{bmatrix} 2 & 3 & 4 & 3 & 4 & 1 & 3 & 4 \end{bmatrix}. \tag{3.1.28}$$

After having identified $L_1 = \delta_4 \begin{bmatrix} 2 & 3 & 4 & 3 \end{bmatrix}$ and $L_2 = \delta_4 \begin{bmatrix} 4 & 1 & 3 & 4 \end{bmatrix}$, one can compute L_{tot} as follows:

$$L_{tot} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}. \tag{3.1.29}$$

It follows that the sets of nodes and edges are respectively:

$$\begin{cases} \mathcal{V} = \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\} \\ \mathcal{E} = \{(\delta_4^1,\, \delta_4^2),\, (\delta_4^1,\, \delta_4^4),\, (\delta_4^2,\, \delta_4^1),\, (\delta_4^2,\, \delta_4^3), \\ \qquad (\delta_4^3,\, \delta_4^3),\, (\delta_4^3,\, \delta_4^4),\, (\delta_4^4,\, \delta_4^3),\, (\delta_4^4,\, \delta_4^4)\}. \end{cases} \tag{3.1.30}$$

The state graph is reported in Fig. 3.1.2, where two communication classes are highlighted. One can identify a closed class $\mathcal{K}_1$:

$$\mathcal{K}_1 = \{\delta_4^3,\, \delta_4^4\}, \tag{3.1.31}$$

and a transient class $\mathcal{K}_2$:

$$\mathcal{K}_2 = \{\delta_4^1,\, \delta_4^2\}. \tag{3.1.32}$$

It is possible to observe that, starting from an initial state of class $\mathcal{K}_2$, the trajectory can remain inside that class as long as the input is suitably chosen. Instead, once the trajectory moves to class $\mathcal{K}_1$, it cannot re-enter the transient class.

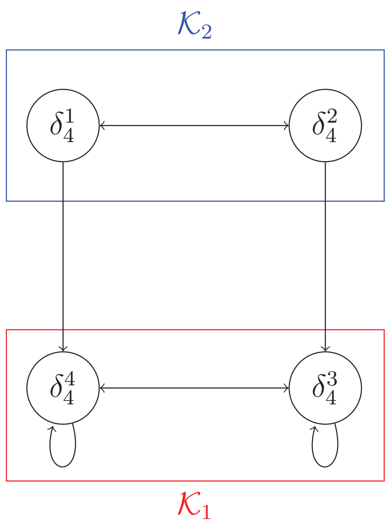

Figure 3.1.2: Communication Classes of a Boolean Control Network

3.2 Reachability

The first section of this chapter showed that the evolution of a B.C.N. cannot be uniquely predetermined as the one of a B.N. due to the presence of inputs. However, the powerfulness of B.C.N. lies in the possibility to control the trajectory to some desired states. An important property that will be now discussed is reachability.

Definition 3.2.1 (Reachability from a State in τ Steps)
Given a B.C.N. as in eqn. (3.1.1), a state $\delta_{2^n}^j \in \mathcal{L}^{2^n}$, with $0 < j \leq 2^n$ and $n \in \mathbb{N}^*$, is reachable in $\tau \in \mathbb{N}$ steps from the state $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i \leq 2^n$, if there exists a sequence of τ inputs $\{\mathbf{u}(0),\, \ldots,\, \mathbf{u}(\tau-1)\}$ such that the trajectory evolves from $\mathbf{x}(0) = \delta_{2^n}^i$ into $\mathbf{x}(\tau) = \delta_{2^n}^j$.

Let us consider an example in this regard.

Example 3.2.1 (Reachability from a State)
Consider a B.C.N. described as in eqn. (3.1.5) with:

$$L = \delta_4 \begin{bmatrix} 2 & 2 & 1 & 3 & 3 & 1 & 4 & 2 \end{bmatrix}. \tag{3.2.1}$$

Then, L_{tot} is as follows:

$$L_{tot} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{3.2.2}$$

and the corresponding state graph is shown in Fig. 3.2.1.
Let us now check the reachability in $\tau = 1$ step. One finds that:

$$\begin{cases} \delta_4^1 \xrightarrow{\delta_2^1} \delta_4^2 \\ \delta_4^1 \xrightarrow{\delta_2^2} \delta_4^3 \\ \\ \delta_4^2 \xrightarrow{\delta_2^1} \delta_4^2 \\ \delta_4^2 \xrightarrow{\delta_2^2} \delta_4^1 \\ \\ \delta_4^3 \xrightarrow{\delta_2^1} \delta_4^1 \\ \delta_4^3 \xrightarrow{\delta_2^2} \delta_4^4 \\ \\ \delta_4^4 \xrightarrow{\delta_2^1} \delta_4^3 \\ \delta_4^4 \xrightarrow{\delta_2^2} \delta_4^2, \end{cases} \tag{3.2.3}$$

where $\xrightarrow{\delta_2^i}$ indicates a state transition in one step under the input δ_2^i, with $i = \{1, 2\}$. From these results, one can conclude, for example, that δ_4^3 is reachable in one step from δ_4^1 for $\mathbf{u}(\cdot) = \delta_2^2$, while δ_4^2 cannot be reached in one step from δ_4^3.

Def. 3.2.1 can be generalised as follows.

Definition 3.2.2 (Reachability from a State)
Given a B.C.N. as in eqn. (3.1.1), a state $\delta_{2^n}^j \in \mathcal{L}^{2^n}$, with $0 < j \leq 2^n$ and $n \in \mathbb{N}^*$, is reachable from the state $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i \leq 2^n$, if there exist τ and one or more sequences of τ inputs $\{\mathbf{u}(0), \ldots, \mathbf{u}(\tau-1)\}$ such that the trajectory evolves from $\mathbf{x}(0) = \delta_{2^n}^i$ into $\mathbf{x}(\tau) = \delta_{2^n}^j$.

Thanks to Defs. 3.2.1 and 3.2.2, it is possible to define the reachable space.

Definition 3.2.3 (Reachable Space)
The set of all the states that are reachable in τ steps from $\delta_{2^n}^i$ is called the reachable space in τ steps from $\delta_{2^n}^i$ and is denoted by $\mathcal{R}_\tau[\delta_{2^n}^i] \subseteq \mathcal{L}^{2^n}$. The set of all the states that are reachable from $\delta_{2^n}^i$ independently of the number of steps is called the reachable space from $\delta_{2^n}^i$, is denoted by $\mathcal{R}[\delta_{2^n}^i] \subseteq \mathcal{L}^{2^n}$ and is computed as follows:

$$\mathcal{R}[\delta_{2^n}^i] = \bigcup_{\tau=0}^{+\infty} \mathcal{R}_\tau[\delta_{2^n}^i]. \tag{3.2.4}$$

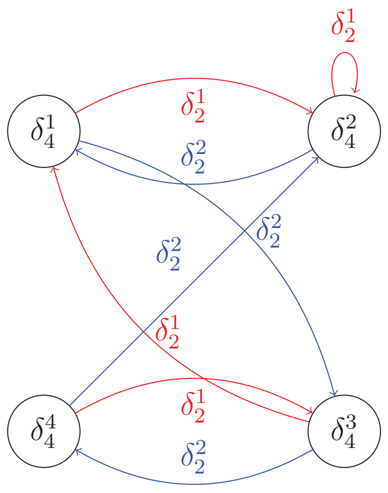

Figure 3.2.1: State Graph of a Boolean Control Network

The following proposition discusses reachability taking into account the dimension of a B.C.N..

Proposition 3.2.1 (Reachability from a State)
Given a B.C.N. with 2^n states, if a state $\delta_{2^n}^j \in \mathcal{L}^{2^n}$ is reachable from $\delta_{2^n}^i \in \mathcal{L}^{2^n}$ in a finite number of steps τ, then $\tau \leq 2^n$.

Due to this proposition, eqn. (3.2.4) can be rewritten as follows:

$$\mathcal{R}[\delta_{2^n}^i] = \bigcup_{\tau=0}^{2^n} \mathcal{R}_\tau[\delta_{2^n}^i]. \tag{3.2.5}$$

Moreover, given two states $\delta_{2^n}^i \in \mathcal{L}^{2^n}$ and $\delta_{2^n}^j \in \mathcal{L}^{2^n}$ that belong to the same communication class (see Def. 2.5.5), it follows that:

$$\begin{cases} \delta_{2^n}^i \in \mathcal{R}[\delta_{2^n}^j] \\ \delta_{2^n}^j \in \mathcal{R}[\delta_{2^n}^i]. \end{cases} \tag{3.2.6}$$

An example is now presented.

Example 3.2.2 (Reachable Space)
By referring to Ex. 3.2.1, the reachable spaces for the various states are:

$$\begin{aligned} \mathcal{R}[\delta_4^1] &= \{\delta_4^1, \delta_4^2, \delta_4^3, \delta_4^4\} \\ \mathcal{R}[\delta_4^2] &= \{\delta_4^1, \delta_4^2, \delta_4^3, \delta_4^4\} \\ \mathcal{R}[\delta_4^3] &= \{\delta_4^1, \delta_4^2, \delta_4^3, \delta_4^4\} \\ \mathcal{R}[\delta_4^4] &= \{\delta_4^1, \delta_4^2, \delta_4^3, \delta_4^4\}. \end{aligned} \tag{3.2.7}$$

In this particular case, each state guarantees global reachability, a concept that will be investigated in the following.

Let us start with the definition of global reachability from a state.

Definition 3.2.4 (Global Reachability from a State)
If the reachable space of the state $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i \leq 2^n$ and $n \in \mathbb{N}^*$, coincides with the whole state space $\mathcal{L}^{2^n}$, that is $\mathcal{R}[\delta_{2^n}^i] = \mathcal{L}^{2^n}$, then the B.C.N. is said to be globally reachable from $\delta_{2^n}^i$.

Example 3.2.3 (Global Reachability from a State)
In Ex. 3.2.2, since the reachable space of each state coincides with the whole state space, one can conclude that global reachability holds for δ_4^i, with $i = \{1, 2, 3, 4\}$. In fact:

$$\mathcal{R}[\delta_4^1] = \mathcal{R}[\delta_4^2] = \mathcal{R}[\delta_4^3] = \mathcal{R}[\delta_4^4] = \{\delta_4^1, \delta_4^2, \delta_4^3, \delta_4^4\} = \mathcal{L}^4. \tag{3.2.8}$$

The matrix L_{tot} proves to be useful for the study of reachability, as shown in the following theorem.

Theorem 3.2.1 (Reachability and L_{tot})
Given a B.C.N. as in eqn. (3.1.5) and its matrix L_{tot} defined as in eqn. (3.1.18), a state $\delta_{2^n}^j$ is reachable in $\tau \in \mathbb{N}$ steps from $\delta_{2^n}^i$, with $0 < i, j \leq 2^n$ and $n \in \mathbb{N}^*$, if and only if

$$[L_{tot}^\tau]_{j,\, i} = 1. \tag{3.2.9}$$

By Thm. 3.2.1, the state $\delta_{2^n}^j$ is reachable from $\delta_{2^n}^i$ if and only if:

$$\left[\bigvee_{\tau=1}^{2^n} L_{tot}^\tau\right]_{j,\, i} = 1. \tag{3.2.10}$$

Moreover, a B.C.N. is globally reachable from $\delta_{2^n}^i$ if and only if

$$\mathrm{Col}_i\left(\bigvee_{\tau=1}^{2^n} L_{tot}^\tau\right) = \mathbf{1}_{2^n}, \tag{3.2.11}$$

where $\mathbf{1}_{2^n}$ represents the unitary column vector of length 2^n.

Example 3.2.4 (Global Reachability)
Recalling Ex. 3.2.1, the logical disjunctions ($\vee$ operator) of the powers of L_{tot}, for $\tau = 1, 2, 3, 4$, are:

$$\begin{gathered} L_{tot} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \bigvee_{\tau=1}^{2} L_{tot}^\tau = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \\ \bigvee_{\tau=1}^{3} L_{tot}^\tau = \bigvee_{\tau=1}^{4} L_{tot}^\tau = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \end{gathered} \tag{3.2.12}$$

It is easy to see that for $\tau \geq 3$ each state can reach all the others because $\mathrm{Col}_i\left(\bigvee_{\tau=1}^{4} L_{tot}^\tau\right) = \mathbf{1}_4$, with $0 < i \leq 4$. Instead, after two steps, only the state δ_4^4 cannot be reached from δ_4^2 since $[L_{tot}^2]_{4,\,2} = 0$. In fact, the only path able to connect these two states has length 3 ($[L_{tot}^3]_{4,\,2} = 1$).

The last comment of the previous example highlights a connection between the concept of reachability and the state graph of a B.C.N.. Saying that a state $\delta_{2^n}^j$ is reachable from $\delta_{2^n}^i$ is equivalent to saying that there exists a directed path from $\delta_{2^n}^i$ to $\delta_{2^n}^j$ in the state graph.

3.3 Controllability

Another important problem in Systems Theory is controllability, which can be considered the problem of reachability from the opposite point of view.

Definition 3.3.1 (Controllability to a State)
Consider a B.C.N. as in eqn. (3.1.5). A state $\delta_{2^n}^i$ is said to be controllable to the state $\delta_{2^n}^j$ if the latter is reachable from the first one. Then, it must hold that:

$$\delta_{2^n}^j \in \mathcal{R}[\delta_{2^n}^i]. \tag{3.3.1}$$

Definition 3.3.2 (Controllable Space)
Given a B.C.N. as in eqn. (3.1.5), the set of all the states $\delta_{2^n}^i \in \mathcal{L}^{2^n}$ which are controllable to $\delta_{2^n}^j \in \mathcal{L}^{2^n}$, with $0 < i,\, j \leq 2^n$, is called controllable space of $\delta_{2^n}^j$ and is denoted by $\mathcal{C}[\delta_{2^n}^j] \subseteq \mathcal{L}^{2^n}$.

Definition 3.3.3 (Global Controllability to a State)
A B.C.N. described as in eqn. (3.1.5) is said to be globally controllable to the state $\delta_{2^n}^j \in \mathcal{L}^{2^n}$ if every $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i \leq 2^n$, is controllable to $\delta_{2^n}^j$. This translates into the following:

$$\delta_{2^n}^j \in \mathcal{R}[\delta_{2^n}^i], \quad \forall\, \delta_{2^n}^i \in \mathcal{L}^{2^n}. \tag{3.3.2}$$

Definition 3.3.4 (Global Controllability)
A B.C.N. described as in eqn. (3.1.5) is said to be globally controllable if each state is reachable from all the other states. This amounts to saying that:

$$\delta_{2^n}^j \in \mathcal{R}[\delta_{2^n}^i], \quad \forall\, \delta_{2^n}^i,\, \delta_{2^n}^j \in \mathcal{L}^{2^n}. \tag{3.3.3}$$

An example is now presented.

Example 3.3.1 (Controllability to a State)
Recalling Ex. 3.2.1, let us verify the controllability in one step:

$$\begin{cases} \delta_4^1 \xleftarrow{\delta_2^2} \delta_4^2 \\ \delta_4^1 \xleftarrow{\delta_2^1} \delta_4^3 \\ \\ \delta_4^2 \xleftarrow{\delta_2^1} \delta_4^1 \\ \delta_4^2 \xleftarrow{\delta_2^1} \delta_4^2 \\ \delta_4^2 \xleftarrow{\delta_2^2} \delta_4^4 \\ \\ \delta_4^3 \xleftarrow{\delta_2^1} \delta_4^1 \\ \delta_4^3 \xleftarrow{\delta_2^1} \delta_4^4 \\ \\ \delta_4^4 \xleftarrow{\delta_2^2} \delta_4^3, \end{cases} \tag{3.3.4}$$

where $\xleftarrow{\delta_2^i}$ indicates a state transition in one step under the input δ_2^i, with $i = \{1,\, 2\}$. From the list above, one can conclude that δ_4^2 is controllable in one step to δ_4^1 with

the input δ_2^2, while δ_4^2 is not controllable in one step to δ_4^3.
Moreover, it is possible to compute the controllable spaces:

$$\begin{aligned}\mathcal{C}[\delta_4^1] &= \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\}\\ \mathcal{C}[\delta_4^2] &= \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\}\\ \mathcal{C}[\delta_4^3] &= \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\}\\ \mathcal{C}[\delta_4^4] &= \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\}.\end{aligned} \tag{3.3.5}$$

Since $\mathcal{C}[\delta_4^i] = \mathcal{L}^4$, the B.C.N. is globally controllable to δ_4^i, with $0 < i \leq 4$. Therefore, the network is globally controllable.

Proposition 3.3.1 (Controllability and L_{tot})
Given a B.C.N. described as in eqn. (3.1.5) with its matrix L_{tot} as in eqn. (3.1.18), a state $\delta_{2^n}^i$ can be controlled in $\tau \in \mathbb{N}$ steps to $\delta_{2^n}^j$, with $0 < i,\, j \leq 2^n$ and $n \in \mathbb{N}^*$, if and only if:

$$[L_{tot}^{\tau}]_{j,\,i} = 1. \tag{3.3.6}$$

Similarly to the consideration drawn for the global reachability, a B.C.N. is globally controllable to $\delta_{2^n}^j$ if and only if:

$$\text{Row}_j \left(\bigvee_{\tau=1}^{2^n} L_{tot}^{\tau} \right) = \mathbf{1}_{2^n}^T, \tag{3.3.7}$$

where $\mathbf{1}_{2^n}^T$ is the unitary row vector of length 2^n.

Example 3.3.2 (Global Controllability)
Let us consider Ex. 3.2.4. It is possible to notice that for $\tau \geq 3$ each state is controllable to all the others because $\text{Row}_j \left(\bigvee_{\tau=1}^{4} L_{tot}^{\tau}\right) = \mathbf{1}_4^T$, with $0 < j \leq 4$. Instead, after two steps, only the state δ_4^2 cannot be controlled to δ_4^4 since $[L_{tot}^2]_{4,\,2} = 0$.

A new matrix, denoted by $\mathbb{L}$, is introduced to simplify the study of controllability of a B.C.N..

Definition 3.3.5 (The matrix $\mathbb{L}$)
Given a B.C.N. as in eqn. (3.1.5), we define the matrix $\mathbb{L}$ as follows:

$$\mathbb{L} := \bigvee_{\tau=0}^{2^n-1} L_{tot}^{\tau}. \tag{3.3.8}$$

Theorem 3.3.1 (Global Controllability and $\mathbb{L}$)
A B.C.N. as in eqn. (3.1.5) is globally controllable if and only if $\mathbb{L}$ in eqn. (3.3.8) has all unitary entries, that is L_{tot} is irreducible.

An example is now shown.

Example 3.3.3 (Global Controllability and $\mathbb{L}$)
Consider Ex. 3.2.4. It has been shown in eqn. (3.2.12) that:

$$\mathbb{L} = \bigvee_{\tau=0}^{3} L_{tot}^{\tau} = \begin{bmatrix} 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{3.3.9}$$

Then, the B.C.N. is globally controllable.

The matrix $\mathbb{L}$ contains the information about which states are reachable and, from the other perspective, which states are controllable. However, it is not possible to retrieve the number of steps that are necessary for a state to be controlled to another state. Therefore, in the following, a new definition of controllability is introduced.

Definition 3.3.6 (Strong Global Controllability)
A B.C.N. is said to be globally controllable in $\tau \in \mathbb{Z}$ steps in the strong sense if:

$$\forall\, \delta_{2^n}^i,\, \delta_{2^n}^j \in \mathcal{L}^{2^n},\, 0 < i,\, j \leq 2^n,\, \exists\, \{\mathbf{u}(0),\, \dots,\, \mathbf{u}(\tau - 1)\} \tag{3.3.10}$$

such that the trajectory moves from $\mathbf{x}(0) = \delta_{2^n}^i$ to $\mathbf{x}(\tau) = \delta_{2^n}^j$.

This definition requires that every state can be controlled to any other possible state in exactly τ steps in order to have a globally controllable in the strong sense B.C.N.. This definition differs from Def. 3.3.4, where it is stated that every state is controllable to any other possible state in a finite number of steps.
In the following theorem a relation between global controllability in the strong sense and the matrix L_{tot} is shown.

Theorem 3.3.2 (Strong Global Controllability and L_{tot})
A B.C.N. described as in eqn. (3.1.5) is globally controllable in the strong sense if and only if its matrix L_{tot} is primitive.

Another important result for global controllability is presented in the following proposition.

Proposition 3.3.2 (Strong Global Controllability)
If a B.C.N. is globally controllable in the strong sense in $\tau \in \mathbb{Z}$ steps, then it is globally controllable in the strong sense $\forall\, k \geq \tau$.

Let us present an example about the matrix $\mathbb{L}$.

Example 3.3.4 (Strong Global Controllability and $\mathbb{L}$)
Consider Ex. 3.2.1. Since the matrix $\mathbb{L}$ is as in eqn. (3.3.9), one can say that the B.C.N. is globally controllable and also globally controllable in the strong sense.

3.4 Stabilisation

This section treats the problem of stabilisation, that is the problem of finding the input values so that the trajectory of a B.C.N. takes only specific values.

Definition 3.4.1 (Stabilisation to a Fixed Point)
A B.C.N. is stabilisable to a fixed point $\delta_{2^n}^i \in \mathcal{L}^{2^n}$, with $0 < i \leq 2^n$ and $n \in \mathbb{N}^*$, if, $\forall\, \mathbf{x}(0) \in \mathcal{L}^{2^n}$, there exists $\tau \in \mathbb{Z}^+$ and an input sequence $\mathbf{u}(t)$, with $t \in \mathbb{Z}^+$, such that $\mathbf{x}(t) = \delta_{2^n}^i$, for every $t \geq \tau$.

A similar definition is now given for the more general case of a limit cycle.

Definition 3.4.2 (Stabilisation to a Limit Cycle)
A B.C.N. is stabilisable to a limit cycle $\mathcal{C} = \{\delta_{2^n}^{i_1},\, \dots,\, \delta_{2^n}^{i_k}\}$, with $0 < i_h \leq 2^n$, $0 < h \leq k$ and $n \in \mathbb{N}^*$, with $\delta_{2^n}^i \neq \delta_{2^n}^j$ for $i \neq j$, if, $\forall\, \mathbf{x}(0) \in \mathcal{L}^{2^n}$, there exists $\tau \in \mathbb{Z}^+$ and an input sequence $\mathbf{u}(t)$, with $t \in \mathbb{Z}^+$, such that $\mathbf{x}(t) = \delta_{2^n}^{i_h}$, for every $t \geq \tau$, with $h = (t - \tau + 1) \bmod k$.

The following proposition characterises the problem of stabilisation to a limit cycle.

Proposition 3.4.1 (Stabilisation to a Limit Cycle)
A B.C.N. is stabilisable to a limit cycle $\mathcal{C} = \{\delta_{2^n}^{i_1}, \ldots, \delta_{2^n}^{i_k}\}$ if and only if these two conditions hold:

- the B.C.N. is globally controllable to $\delta_{2^n}^{i_h} \in \mathcal{C}$, with $0 < i_h \leq 2^n$ and $0 < h \leq k$;
- for every (i_l, i_{l+1}), with $l \in [1, k]$ and $i_{k+1} = i_1$, there exists an input $\mathbf{u}(t) = \delta_{2^m}^{j_l}$ such that $\delta_{2^m}^{i_{l+1}} = \mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^m}^{j_l} \ltimes \delta_{2^n}^{i_l} = L_{j_l} \ltimes \delta_{2^n}^{i_l}$.

This proposition holds also in the case of a fixed point since the latter is a limit cycle of length one. The two previous conditions are slightly simplified:

- $\delta_{2^n}^{i}$ is reachable from every initial condition $\mathbf{x}(0)$, that is $\delta_{2^n}^{i} \in \mathcal{R}^*$;
- there exists $\delta_{2^m}^{k}$, with $0 < k \leq 2^m$, such that $\delta_{2^n}^{i} = \mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = L \ltimes \delta_{2^m}^{k} \ltimes \delta_{2^n}^{i} = L_k \ltimes \delta_{2^n}^{i}$.

An example of stabilisation of a B.C.N. is now proposed.

Example 3.4.1 (Stabilisation to a Limit Cycle)
Given a B.C.N. as in eqn. (3.1.5) with two state variables and one input with:

$$L = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 3 & 3 & 3 \end{bmatrix}, \tag{3.4.1}$$

where $L_1 = \begin{bmatrix} 2 & 2 & 1 & 1 \end{bmatrix}$ and $L_2 = \begin{bmatrix} 1 & 3 & 3 & 3 \end{bmatrix}$, a possible limit cycle of the B.C.N. is:

$$\mathcal{C} = \{\delta_4^1, \delta_4^2, \delta_4^3\}, \tag{3.4.2}$$

because, if $\mathbf{x}(t) = \delta_4^1$ with $t \in \mathbb{Z}$, one can verify that:

$$\begin{cases} \mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = L \ltimes \delta_2^1 \ltimes \delta_4^1 = \delta_4^2 \\ \mathbf{x}(t+2) = L \ltimes \mathbf{u}(t+1) \ltimes \mathbf{x}(t+1) = L \ltimes \delta_2^2 \ltimes \delta_4^2 = \delta_4^3 \\ \mathbf{x}(t+3) = L \ltimes \mathbf{u}(t+2) \ltimes \mathbf{x}(t+2) = L \ltimes \delta_2^1 \ltimes \delta_4^3 = \delta_4^1, \end{cases} \tag{3.4.3}$$

satisfying the second condition of Prop. 3.4.1. Now, the global controllability to a state of $\mathcal{C}$ of the B.C.N. needs to be verified in order to verify that it is stabilisable. Therefore, let us compute the matrix $\mathbb{L}$ as in eqn. (3.3.8):

$$\mathbb{L} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4.4}$$

where it holds that:

$$\mathrm{Row}_1[\mathbb{L}] = \mathrm{Row}_2[\mathbb{L}] = \mathrm{Row}_3[\mathbb{L}] = \mathbf{1}_4, \tag{3.4.5}$$

This proves the global controllability of the B.C.N. to the states belonging to $\mathcal{C}$. The corresponding state graph is reported in Fig. 3.4.1.

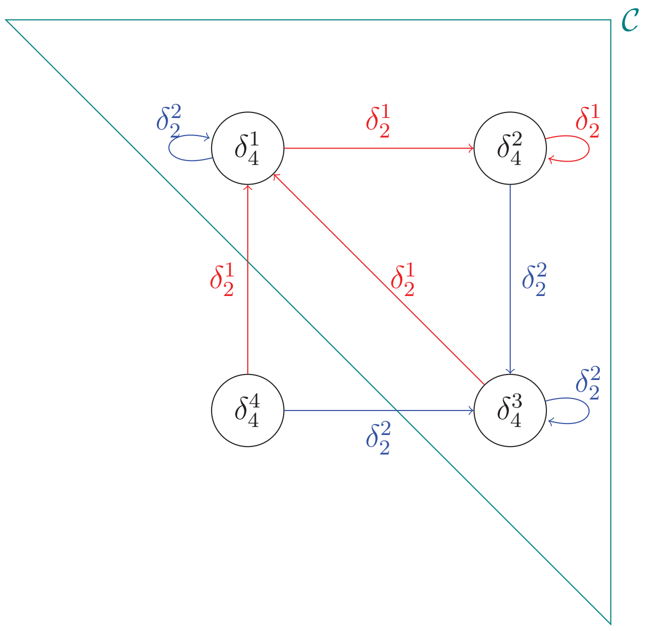

Figure 3.4.1: Stabilisation of a Boolean Control Network

3.5 Feedback Control

In the previous section, the inputs that stabilised the B.C.N. were chosen arbitrarily at every time instant. Now, the problem of stabilisation is solved using a time-invariant feedback law, where the input vector is a linear combination of state vector entries:

$$\mathbf{u}(t) = K \ltimes \mathbf{x}(t), \tag{3.5.1}$$

with

$$K = \begin{bmatrix} \delta_{2^m}^{k_1} & \delta_{2^m}^{k_2} & \cdots & \delta_{2^m}^{k_n} \end{bmatrix} \in \mathcal{L}^{2^m \times 2^n}. \tag{3.5.2}$$

This concept is formalized in the following proposition.

Proposition 3.5.1 (Stabilisation by means of a State Feedback Law)
If a B.C.N. is stabilisable to a limit cycle $\mathcal{C} = \{\delta_{2^n}^{i_1}, \ldots, \delta_{2^n}^{i_k}\}$, then it can be stabilised by means of a state feedback law.

In the following, two examples of feedback stabilisation are presented.

Example 3.5.1 (Feedback Stabilisation to a Fixed Point)
Consider a B.C.N. with three state variables and one input. The transition matrix is:

$$L = \begin{bmatrix} L_1 & L_2 \end{bmatrix}, \tag{3.5.3}$$

where

$$\begin{cases} L_1 = \begin{bmatrix} 2 & 2 & 2 & 8 & 4 & 8 & 5 & 2 \end{bmatrix} \\ L_2 = \begin{bmatrix} 1 & 1 & 4 & 6 & 2 & 5 & 6 & 8 \end{bmatrix}. \end{cases} \tag{3.5.4}$$

The state graph of this B.C.N. is reported in Fig. 3.5.1. It is easy to notice either from the matrix L_2 or from the state graph that δ_8^2 is a fixed point which the network is globally controllable to. Since both points of Prop. 3.4.1 are satisfied for this fixed

point, one can conclude that the B.C.N. is stabilisable to the state δ_8^2.
Let us now compute the parent states (see Def. 2.5.2) of δ_8^2 as follows:

$$\begin{cases} L^{-0}[\delta_8^2] = \{\delta_8^2\} \\ L^{-1}[\delta_8^2] = \{\delta_8^1,\, \delta_8^3,\, \delta_8^5,\, \delta_8^8\} \\ L^{-2}[\delta_8^2] = \{\delta_8^4,\, \delta_8^6,\, \delta_8^7\} \\ L^{-3}[\delta_8^2] = \emptyset \\ L^{-4}[\delta_8^2] = \emptyset \\ L^{-5}[\delta_8^2] = \emptyset \\ L^{-6}[\delta_8^2] = \emptyset \\ L^{-7}[\delta_8^2] = \emptyset. \end{cases} \tag{3.5.5}$$

The input that keeps the trajectory of the B.C.N. at the fixed point is δ_2^1. Then, the second column of K is the logic vector δ_2^1. Algebraically, it holds that:

$$\delta_8^2 \xrightarrow{\delta_2^1} \delta_8^2 \Longrightarrow \mathrm{Col}_2(K) = \delta_2^1, \tag{3.5.6}$$

where $\xrightarrow{\delta_2^1}$ indicates a state transition in one step under the input $\mathbf{u}(\cdot) = \delta_2^1$. Every other column of the feedback matrix K is found assigning the value of the input that brings a parent state one step closer to the fixed point. In the case of a parent state of length one, the right input guarantees to reach the fixed point after the transition. It is important to remark that the input is not necessarily unique. So, there can exist multiple matrices K that stabilise the trajectory to a specific fixed point.
The other columns of K are filled as follows:

$$\begin{cases} \delta_8^1 \in L^{-1}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^2 \in L^{-0}[\delta_8^2] \Longrightarrow \mathrm{Col}_1(K) = \delta_2^1 \\ \delta_8^3 \in L^{-1}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^2 \in L^{-0}[\delta_8^2] \Longrightarrow \mathrm{Col}_3(K) = \delta_2^1 \\ \delta_8^5 \in L^{-1}[\delta_8^2] \xrightarrow{\delta_2^2} \delta_8^2 \in L^{-0}[\delta_8^2] \Longrightarrow \mathrm{Col}_5(K) = \delta_2^2 \\ \delta_8^8 \in L^{-1}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^2 \in L^{-0}[\delta_8^2] \Longrightarrow \mathrm{Col}_8(K) = \delta_2^1 \\ \delta_8^4 \in L^{-2}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^8 \in L^{-1}[\delta_8^2] \Longrightarrow \mathrm{Col}_4(K) = \delta_2^1 \\ \delta_8^6 \in L^{-2}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^8 \in L^{-1}[\delta_8^2] \Longrightarrow \mathrm{Col}_6(K) = \delta_2^1 \\ \delta_8^7 \in L^{-2}[\delta_8^2] \xrightarrow{\delta_2^1} \delta_8^5 \in L^{-1}[\delta_8^2] \Longrightarrow \mathrm{Col}_7(K) = \delta_2^1. \end{cases} \tag{3.5.7}$$

Using Prop. 3.5.1 and reordering the columns found above, the feedback matrix becomes:

$$K = \delta_2 \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \end{bmatrix}. \tag{3.5.8}$$

Example 3.5.2 (Feedback Stabilisation to a Limit Cycle)

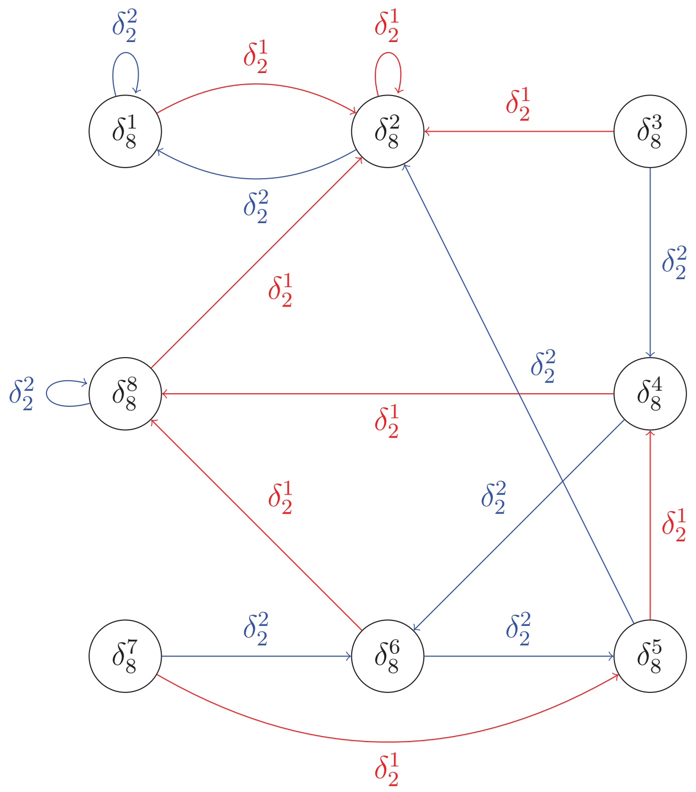

Figure 3.5.1: Feedback Stabilisation of a Boolean Control Network

In the B.C.N. of Ex. 3.5.1 the following five limit cycles are present:

$$\begin{cases} \mathcal{C}_1 : \delta_8^1 \xrightarrow{\delta_2^2} \delta_8^1 \\ \mathcal{C}_2 : \delta_8^1 \xrightarrow{\delta_2^1} \delta_8^2 \xrightarrow{\delta_2^2} \delta_8^1 \\ \mathcal{C}_3 : \delta_8^2 \xrightarrow{\delta_2^1} \delta_8^2 \\ \mathcal{C}_4 : \delta_8^4 \xrightarrow{\delta_2^2} \delta_8^6 \xrightarrow{\delta_2^2} \delta_8^5 \xrightarrow{\delta_2^1} \delta_8^4 \\ \mathcal{C}_5 : \delta_8^8 \xrightarrow{\delta_2^2} \delta_8^8. \end{cases} \tag{3.5.9}$$

The only states that the B.C.N. is globally controllable to are:

$$\mathcal{R}^* = \{\delta_8^1,\, \delta_8^2\}. \tag{3.5.10}$$

Then, by Prop. 3.4.1, the B.C.N. is stabilisable to the following cycles:

$$\begin{cases} \mathcal{C}_1 : \delta_8^1 \xrightarrow{\delta_2^2} \delta_8^1 \\ \mathcal{C}_2 : \delta_8^1 \xrightarrow{\delta_2^1} \delta_8^2 \xrightarrow{\delta_2^2} \delta_8^1 \\ \mathcal{C}_3 : \delta_8^2 \xrightarrow{\delta_2^1} \delta_8^2. \end{cases} \tag{3.5.11}$$

By applying Prop. 3.5.1, the feedback matrices for the cycles are as follows:

$$\begin{cases} \mathcal{C}_1 : K_1 = \delta_2 \begin{bmatrix} 2 & 2 & 1 & 1 & 2 & 1 & 1 & 1 \end{bmatrix} \\ \mathcal{C}_2 : K_2 = \delta_2 \begin{bmatrix} 1 & 2 & 1 & 1 & 2 & 1 & 1 & 1 \end{bmatrix} \\ \mathcal{C}_3 : K_3 = \delta_2 \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \end{bmatrix}. \end{cases} \tag{3.5.12}$$

By substituting eqn. (3.5.1) of feedback control into eqn. (3.1.5) and by performing simple manipulations, it is possible to transform the original B.C.N. into the following B.N.:

$$\begin{aligned} \mathbf{x}(t+1) &= L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) = \\ &= L \ltimes K \ltimes \mathbf{x}(t) \ltimes \mathbf{x}(t) = \\ &= L \ltimes K \ltimes \mathbf{x}^2(t) = \\ &= L \ltimes K \ltimes M_r \ltimes \mathbf{x}(t) = \\ &= L_K \ltimes \mathbf{x}(t), \end{aligned} \tag{3.5.13}$$

where $M_r = \delta_{2^n \cdot 2^n} \begin{bmatrix} 1 & 2^n+2 & 2\cdot 2^n+3 & \cdots & (2^n-1)\cdot 2^n + 2^n \end{bmatrix} \in \mathcal{L}^{2^n\cdot 2^n \times 2^n}$ is the power reduction matrix of eqn. (1.3.28) for $\mathbf{x}(\cdot) \in \mathcal{L}^{2^n}$ and $L_K = L \ltimes K \ltimes M_r \in \mathcal{L}^{2^n \times 2^n}$ is the structure matrix of the new B.N..
In particular:

$$L_K := \begin{bmatrix} \mathrm{Col}_1(L_{k_1}) & \mathrm{Col}_2(L_{k_2}) & \cdots & \mathrm{Col}_n(L_{k_n}) \end{bmatrix}. \tag{3.5.14}$$

Then, a B.C.N. can be stabilised to the limit cycle $\mathcal{C}$ by means of state feedback. The stabilised B.C.N. is equivalent to a B.N. whose only attractor is $\mathcal{C}$, as shown in eqn. (3.5.13).
An example of this last comment is now proposed.

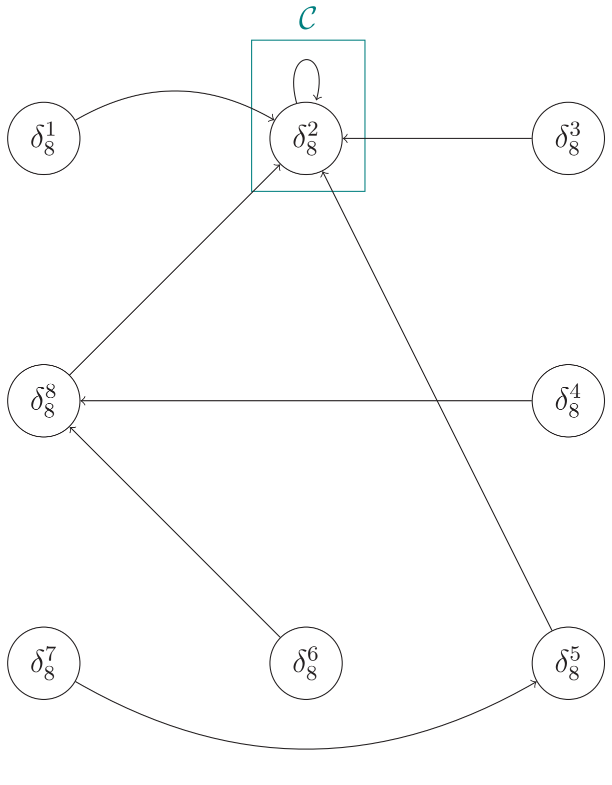

Figure 3.5.2: Equivalent Boolean Network

Example 3.5.3 (Equivalent Boolean Network)
By referring again to Ex. 3.5.1, the matrix L_K is the following one:

$$L_K = L \ltimes K \ltimes M_r = \delta_8 \begin{bmatrix} 2 & 2 & 2 & 8 & 2 & 8 & 5 & 2 \end{bmatrix}. \tag{3.5.15}$$

The state graph of the resulting B.N. is reported in Fig. 3.5.2, where the stable fixed point obtained by means of state feedback is highlighted.

The references examined to write this chapter are the following: [13], [14], [15], [19], [20], [21], [22].

Chapter 4

Probabilistic Boolean Networks

This chapter focuses on Probabilistic Boolean Networks (P.B.N.s), which are a stochastic extension of the B.N.s of the previous chapters. In fact, a P.B.N. can be considered as a collection of B.N.s and the choice of a specific B.N. is subjected to a probability. Then, P.B.N.s share the property of B.N.s of representing dependencies between genes but are able to cope with uncertainties, both in the data and the model selection.

4.1 Probabilistic Boolean Networks Dynamics

The structure of a probabilistic boolean network is defined in the following.

Definition 4.1.1 (Probabilistic Boolean Network)
A P.B.N. is a dynamical system involving boolean state variables that can influence each other by means of logic functions. The state variables are $\chi_1(t)$, $\chi_2(t)$, ..., $\chi_n(t) \in \mathcal{B}$, with $n \in \mathbb{N}^*$. The evolution of the entire network is determined by a set of n logic first order difference equations:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t)\right) \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t)\right) \\ \vdots \\ \chi_n(t+1) = f_n\left(\chi_1(t),\, \chi_2(t),\, \ldots,\, \chi_n(t)\right), \end{cases} \tag{4.1.1}$$

where, at every time instant t, each logic function f_i is selected from a collection of $l_i < \infty$ possible models, namely $f_i \in \{f_i^1,\, f_i^2,\, \ldots,\, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \to \mathcal{B}$, according to the probability of f_i being $f_i^{\gamma_i}$, i.e. $\Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, with $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$, $\gamma_i \in \{1,\, 2,\, \ldots,\, l_i\}$ and $i \in \{1,\, 2,\, \ldots,\, n\}$.

An example of probabilistic boolean network is presented next.

Example 4.1.1 (Probabilistic Boolean Network)
Let χ_1, χ_2 and $\chi_3 \in \mathcal{B}$ be the state variables and let f_1, f_2 and $f_3 : \mathcal{B}^3 \to \mathcal{B}$ be the function sets, where $f_1 = \{f_1^1,\, f_1^2\}$, $f_2 = \{f_2^1,\, f_2^2\}$ and $f_3 = \{f_3^1\}$. Then, a possible

system of equations describing the P.B.N. is as follows:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t)\right) = \\ \qquad = \begin{cases} f_1^1\left(\chi_1(t),\, \chi_2(t)\right) = \chi_1(t) \vee \chi_2(t), & p_1^1 = \Pr\{f_1 = f_1^1\} = 0.4 \\ f_1^2\left(\chi_1(t),\, \chi_2(t)\right) = \chi_1(t) \wedge \chi_2(t), & p_1^2 = \Pr\{f_1 = f_1^2\} = 0.6 \end{cases} \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_3(t)\right) = \\ \qquad = \begin{cases} f_2^1\left(\chi_1(t),\, \chi_3(t)\right) = \chi_1(t) \wedge \neg\chi_3(t), & p_2^1 = \Pr\{f_2 = f_2^1\} = 0.5 \\ f_2^2\left(\chi_1(t),\, \chi_3(t)\right) = \chi_1(t) \vee \neg\chi_3(t), & p_2^2 = \Pr\{f_2 = f_2^2\} = 0.5 \end{cases} \\ \chi_3(t+1) = f_3\left(\chi_2(t),\, \chi_3(t)\right) = \\ \qquad = f_3^1\left(\chi_2(t),\, \chi_3(t)\right) = \neg\chi_2(t) \wedge \chi_3(t), \quad p_3^1 = \Pr\{f_3 = f_3^1\} = 1. \end{cases} \tag{4.1.2}$$

Just like in the case of a B.N., a P.B.N. is a logic system that can be converted into its algebraic form using the procedures explained in Chapter 1. This is formally expressed in the following proposition.

Proposition 4.1.1 (Probabilistic Boolean Networks in Algebraic Form)

$$\begin{cases} x_1(t+1) = f_1(x_1(t),\, x_2(t),\, \ldots,\, x_n(t)) = M_{f_1} \ltimes \mathbf{x}(t) \\ x_2(t+1) = f_2(x_1(t),\, x_2(t),\, \ldots,\, x_n(t)) = M_{f_2} \ltimes \mathbf{x}(t) \\ \vdots \\ x_n(t+1) = f_n(x_1(t),\, x_2(t),\, \ldots,\, x_n(t)) = M_{f_n} \ltimes \mathbf{x}(t), \end{cases} \tag{4.1.3}$$

where $x_i(\cdot) \in \mathcal{L}^2$, $f_j(\cdot) : \mathcal{L}^{2^n} \to \mathcal{L}^2$, $M_{f_j} \in \mathcal{L}^{2\times 2^n}$, with $M_{f_j} \in \{M_{f_j}^1,\, M_{f_j}^2,\, \ldots,\, M_{f_j}^{l_j}\}$, $i,\, j \in \{1,\, n\}$, and $\mathbf{x}(t) = \ltimes_{i=1}^n x_i(t) \in \mathcal{L}^{2^n}$.
To summarize this expression, one can rewrite the P.B.N. as:

$$\mathbf{x}(t+1) = L_\gamma \ltimes \mathbf{x}(t), \tag{4.1.4}$$

where $L_\gamma \in \mathcal{L}^{2^n \times 2^n}$ is one of the structure matrices with $\gamma = 1,\, 2,\, \ldots,\, N$ and $N = \prod_{i=1}^n l_i$ denoting the number of possible combinations of logic functions, that is the number of networks. In fact, the probability for each model Σ_γ to be active is:

$$P_\gamma = \Pr\{\text{network } \Sigma_\gamma \text{ is selected}\} = \Pr\{f_1 = f_1^{\gamma_1},\, f_2 = f_2^{\gamma_2},\, \ldots,\, f_n = f_n^{\gamma_n}\}. \tag{4.1.5}$$

In this thesis, the next proposition is always true.

Proposition 4.1.2 (Independent P.B.N.)
A P.B.N. as in eq. (4.1.3) is considered to be independent if the functions $f_1,\, f_2,\, \ldots,$ f_n are independent. Therefore, eq. (4.1.5) can be rewritten as follows:

$$\begin{aligned} P_\gamma &= \Pr\{f_1 = f_1^{\gamma_1},\, f_2 = f_2^{\gamma_2},\, \ldots,\, f_n = f_n^{\gamma_n}\} = \\ &= \Pr\{f_1 = f_1^{\gamma_1}\} \cdot \Pr\{f_2 = f_2^{\gamma_2}\} \cdot \cdots \cdot \Pr\{f_n = f_n^{\gamma_n}\} = \\ &= \prod_{i=1}^n p_i^{\gamma_i}, \quad \gamma = 1,\, 2,\, \ldots,\, N. \end{aligned} \tag{4.1.6}$$

An example is now shown for clarity.

Example 4.1.2 (Probabilistic Boolean Networks in Algebraic Form)
Consider Ex. 4.1.1. Using Thm. 1.4.1, the P.B.N. becomes:

$$\begin{cases} x_1(t+1) & = M_\vee \ltimes D_f^{4,2} \ltimes \mathbf{x}(t) = M_{f_1}^1 \ltimes \mathbf{x}(t) \\ & = M_\wedge \ltimes D_f^{4,2} \ltimes \mathbf{x}(t) = M_{f_1}^2 \ltimes \mathbf{x}(t) \\ x_2(t+1) & = M_\wedge \ltimes (I_2 \otimes M_\neg) \ltimes D_f^{2,2} \ltimes \mathbf{x}(t) = M_{f_2}^1 \ltimes \mathbf{x}(t) \\ & = M_\vee \ltimes (I_2 \otimes M_\neg) \ltimes D_f^{2,2} \ltimes \mathbf{x}(t) = M_{f_2}^2 \ltimes \mathbf{x}(t) \\ x_3(t+1) & = M_\wedge \ltimes M_\neg \ltimes D_r^{2,4} \ltimes \mathbf{x}(t) = M_{f_3}^1 \ltimes \mathbf{x}(t), \end{cases} \tag{4.1.7}$$

where:

$$\begin{cases} M_{f_1}^1 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \\ M_{f_1}^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\ M_{f_2}^1 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \\ M_{f_2}^2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \\ M_{f_3}^1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \ltimes \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}. \end{cases} \tag{4.1.8}$$

Finally, one can compute the $N = l_1 \cdot l_2 \cdot l_3 = 2 \cdot 2 \cdot 1 = 4$ structure matrices of the network by means of the Khatri-Rao product:

$$\begin{aligned} L_1 = M_{f_1}^1 * M_{f_2}^1 * M_{f_3}^1 &= \delta_8 \begin{bmatrix} 4 & 2 & 3 & 2 & 4 & 4 & 7 & 8 \end{bmatrix}, \\ \text{with } P_1 = \Pr\{\Sigma_1 \text{ selected}\} &= p_1^1 \cdot p_2^1 \cdot p_3^1 = 0.2 \\ L_2 = M_{f_1}^1 * M_{f_2}^2 * M_{f_3}^1 &= \delta_8 \begin{bmatrix} 2 & 2 & 1 & 2 & 4 & 2 & 7 & 6 \end{bmatrix}, \\ \text{with } P_2 = \Pr\{\Sigma_2 \text{ selected}\} &= p_1^1 \cdot p_2^2 \cdot p_3^1 = 0.2 \\ L_3 = M_{f_1}^2 * M_{f_2}^1 * M_{f_3}^1 &= \delta_8 \begin{bmatrix} 4 & 2 & 7 & 6 & 8 & 8 & 7 & 8 \end{bmatrix}, \\ \text{with } P_3 = \Pr\{\Sigma_3 \text{ selected}\} &= p_1^2 \cdot p_2^1 \cdot p_3^1 = 0.3 \\ L_4 = M_{f_1}^2 * M_{f_2}^2 * M_{f_3}^1 &= \delta_8 \begin{bmatrix} 2 & 2 & 5 & 6 & 8 & 6 & 7 & 6 \end{bmatrix}, \\ \text{with } P_4 = \Pr\{\Sigma_4 \text{ selected}\} &= p_1^2 \cdot p_2^2 \cdot p_3^1 = 0.3. \end{aligned} \tag{4.1.9}$$

4.1.1 Probabilistic Boolean Networks as Switched Systems

With the possibility that the update function of each state of a B.N. is not unique, the state graph of a P.B.N. changes depending on the selected function for each

state. Therefore, it is possible to interpret a P.B.N. as a boolean switched system that varies among N different boolean networks.
In this case, the factor that makes the system switch is not deterministic, as the value of the inputs for a B.C.N., but is rather the probability that a specific model Σ_γ is selected at time $t \in \mathbb{Z}$. Algebraically, it holds:

$$\mathbf{x}(t+1) = L_\gamma(t) \ltimes \mathbf{x}(t), \tag{4.1.10}$$

where $L_\gamma \in \mathcal{L}^{2^n \times 2^n}$ is the structure matrix of the active B.N. at time t. L_γ is selected by the probability $P_\gamma = \Pr\{\Sigma_\gamma \text{ selected}\}$, with $0 < \gamma \leq N$. Therefore, the P.B.N. can be written as a switched system:

$$\mathbf{x}(t+1) = L_{\sigma(t)} \ltimes \mathbf{x}(t), \tag{4.1.11}$$

where $\sigma(t)$ is the active model taking values in $\{1, \ldots, N\}$ and $P_\gamma = \Pr\{\sigma(t) = \gamma\}$. Then, the P.B.N. can be thought of as:

$$\mathbf{x}(t+1) = L \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \tag{4.1.12}$$

where L is rewritten as:

$$L := \sum_{\gamma=1}^{N} P_\gamma L_\gamma, \tag{4.1.13}$$

and the P.B.N. is driven by the stochastic input sequence $\mathbf{u}(t) \in \mathcal{L}^N$, with $\Pr\{\mathbf{u}(t) = \delta_N^\gamma\} = P_\gamma$.

Let us now consider an example.

Example 4.1.3 (Probabilistic Boolean Networks as Switched Systems)
Consider the same P.B.N. as in Ex. 4.1.2. Since there are two possible update functions for state x_1, two for state x_2 and one for state x_3, i.e. $f_1 = \{f_1^1, f_1^2\}$, $f_2 = \{f_2^1, f_2^2\}$ and $f_3 = \{f_3^1\}$, it means that there are $N = l_1 \cdot l_2 \cdot l_3 = 4$ sub-networks represented by the matrix L as in eq. (4.1.13), where:

$$L = \begin{bmatrix} 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0.4 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.4 & 0.2 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0.3 & 0 & 0.5 \\ 0 & 0 & 0.3 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 0.3 & 0 & 0.5 \end{bmatrix}, \tag{4.1.14}$$

and the different L_γ for $\gamma = 1, 2, 3, 4$ are those in eq. (4.1.9).

The following definitions give an insight on the different networks of a P.B.N..

Definition 4.1.2 (Matrix K of possible models)
Given a P.B.N. as in eq. (4.1.4), the index set of possible models Σ_γ, with $\gamma =$

1, 2, ..., N, is denoted by matrix K as follows:

$$K = \begin{bmatrix} 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 1 & l_n \\ 1 & 1 & \cdots & 2 & 1 \\ 1 & 1 & \cdots & 2 & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \cdots & 2 & l_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ l_1 & l_2 & \cdots & l_{n-1} & l_n \end{bmatrix}, \tag{4.1.15}$$

where $K \in \mathbb{N}^{N \times n}$ and $N = \prod_{i=1}^{n} l_i$. Each row of K represents a possible network with probability $P_\gamma = \Pr\{\text{network } \Sigma_\gamma \text{ is selected}\} = \prod_{i=1}^{n} p_i^{[K]_{\gamma,\, i}}$ where $[K]_{\gamma,\, i}$ is the $(\gamma,\, i)$-th entry of the matrix K.

Definition 4.1.3 (State Transition Matrix A)
Given a P.B.N. as in eq. (4.1.4), let us define the probability of transitioning from a state $(x_1,\, x_2,\, \ldots,\, x_n) \sim \delta_{2^n}^i \in \mathcal{L}^{2^n}$ to the state $(x_1',\, x_2',\, \ldots,\, x_n') \sim \delta_{2^n}^j \in \mathcal{L}^{2^n}$ as follows:

$$\begin{aligned} &\Pr\{(x_1,\, x_2,\, \ldots,\, x_n) \to (x_1',\, x_2',\, \ldots,\, x_n')\} = \\ &= \sum_{\gamma:\, f_1^{K_{\gamma 1}}(x_1, x_2, \ldots, x_n) = x_1',\, f_2^{K_{\gamma 2}}(x_1, x_2, \ldots, x_n) = x_2',\, \ldots,\, f_n^{K_{\gamma n}}(x_1, x_2, \ldots, x_n) = x_n'} P_\gamma = \\ &= \sum_{\gamma=1}^{N} P_\gamma \left[\prod_{i=1}^{n} (1 - |f_i^{K_{\gamma i}}(x_1,\, x_2,\, \ldots,\, x_n) - x_i'|) \right] = \\ &= \sum_{\gamma=1}^{N} \Pr\{(x_1,\, x_2,\, \ldots,\, x_n) \to (x_1',\, x_2',\, \ldots,\, x_n') |\, \text{network } \Sigma_\gamma \text{ is selected}\} \cdot P_\gamma. \end{aligned} \tag{4.1.16}$$

Then, the entries of the state transition matrix A are found as follows:

$$[A]_{i,\, j} = \Pr\{(x_1,\, x_2,\, \ldots,\, x_n) \sim \delta_{2^n}^i \to (x_1',\, x_2',\, \ldots,\, x_n') \sim \delta_{2^n}^j\}, \quad \forall i,\, j = 1,\, 2,\, \ldots,\, 2^n. \tag{4.1.17}$$

It is also possible to state that:

$$\sum_{j=1}^{2^n} [A]_{i,\, j} = 1, \quad \forall i = 1,\, 2,\, \ldots,\, 2^n, \tag{4.1.18}$$

because any state of the P.B.N. is supposed to evolve with probability one to another state during a transition.

Remark 4.1.1
The state transition matrix A is the transpose of the structure matrix L defined in eq. (4.1.13):

$$A = L^T. \tag{4.1.19}$$

χ_1	χ_2	χ_3	f_1^1	f_1^2	f_2^1	f_2^2	f_3^1
0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	1	0	0	0	0
1	0	0	1	0	1	1	0
1	0	1	1	0	0	1	1
1	1	0	1	1	1	1	0
1	1	1	1	1	0	1	0
$p_i^{\gamma_i}$			0.4	0.6	0.5	0.5	1

Table 4.1.1: Truth Table of a Probabilistic Boolean Network

The following example clarifies the procedure to build the transition matrix A.

Example 4.1.4 (State Transition Matrix A)
Consider again Ex. 4.1.1. From the system of equations (4.1.2), it is possible to derive the truth table of the P.B.N., reported in Tab. 4.1.1. Since there are two possible functions for state χ_1, two functions for state χ_2 and one function for state χ_3, the matrix K can be built as follows:

$$K = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 2 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}, \tag{4.1.20}$$

Finally, the state transition matrix A is:

$$\begin{aligned} A = &\begin{bmatrix} 0 & P_2+P_4 & 0 & P_1+P_3 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_2 & 0 & P_1 & 0 & P_4 & 0 & P_3 & 0 \\ 0 & P_1+P_2 & 0 & 0 & 0 & P_3+P_4 & 0 & 0 \\ 0 & 0 & 0 & P_1+P_2 & 0 & 0 & 0 & P_3+P_4 \\ 0 & P_2 & 0 & P_1 & 0 & P_4 & 0 & P_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_2+P_4 & 0 & P_1+P_3 \end{bmatrix} = \\ = &\begin{bmatrix} 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.2 & 0 & 0.3 & 0 & 0.3 & 0 \\ 0 & 0.4 & 0 & 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.4 & 0 & 0 & 0 & 0.6 \\ 0 & 0.2 & 0 & 0.2 & 0 & 0.3 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0.5 \end{bmatrix}, \end{aligned} \tag{4.1.21}$$

using the values of P_γ for $\gamma \in \{1, 2, 3, 4\}$ found in eq. (4.1.9).
Let us now clarify the meaning of this matrix by analysing the entry $[A]_{1,2}$. This corresponds to the transition probability $\Pr\{(1, 1, 1) \to (1, 1, 0)$. Observing the row related to $(\chi_1, \chi_2, \chi_3) = (1, 1, 1)$ in Tab. 4.1.1, one can see that the possible

combinations of the functions that bring the state to the values (1, 1, 0) are either (f_1^1, f_2^2, f_3^1) or (f_1^2, f_2^2, f_3^1). In the matrix K, these combinations correspond to the second and fourth rows which represent respectively that, in the P.B.N., the model Σ_2 is selected with probability P_2 and the model Σ_4 is selected with probability P_4. That is why $[A]_{1,\,2} = P_2 + P_4 = 0.5$.
Let us now concentrate on the entry $[A]_{7,\,7}$. In this case, one can immediately write $[A]_{7,\,7} = 1$ because $[L_\gamma]_{7,\,7} = 1$, so in each transition matrix of eq. (4.1.9) the state (0, 0, 1) $\sim \delta_8^7$ is a fixed point.
It is also possible to prove that eq. (4.1.18) is true by checking matrix A directly.

Theorem 4.1.1 (Probabilistic Boolean Network Nodes Dynamics)
Let $L_\gamma(t)$ be the structure matrix at time t in the algebraic form of eq. (4.1.4). The dynamical behaviour of the $n \in \mathbb{N}^*$ variables of the P.B.N. can be described as follows:

$$\begin{aligned} \mathbf{x}(t) &= L_\gamma(t-1) \ltimes \mathbf{x}(t-1) = \\ &= L_\gamma(t-1) \ltimes L_\gamma(t-2) \ltimes \mathbf{x}(t-2) = \\ &\vdots \\ &= L_\gamma(t-1) \ltimes L_\gamma(t-2) \ltimes \cdots \ltimes L_\gamma(0) \ltimes \mathbf{x}(0). \end{aligned} \tag{4.1.22}$$

From eq. (2.1.3), it is true that $\forall\, i = 1, \ldots, n$:

$$x_i(t) = M_{f_i} \ltimes \mathbf{x}(t-1), \tag{4.1.23}$$

then, also in the case of a P.B.N.:

$$x_i(t) = M_{f_i} \ltimes L_\gamma(t-2) \ltimes \mathbf{x}(t-2). \tag{4.1.24}$$

Since a B.N. described by eq. (2.1.3) is time-invariant and deterministic, given an initial state condition $\mathbf{x}(0) \in \mathcal{L}^{2^n}$, the state trajectory $\mathbf{x}(t)$, with $t \in \mathbb{Z}$, is univocally determined. The structure matrix L, in fact, is unique and does not change over time. From an initial condition $\mathbf{x}(0)$, then, the state trajectory can be pre-determined using eq. (2.1.10).
In a B.N. described as in eq. (2.1.3) which is time-invariant and deterministic, the evolution of the state $\mathbf{x}(t)$, with $t \in \mathbb{Z}$, is univocally determined, starting from any initial condition $\mathbf{x}(0) \in \mathcal{L}^{2^n}$. This is not true for a P.B.N. since the active network Σ_γ at time t depends on the probability P_γ. Therefore, unlike in a B.N., the structure matrix L_γ is not unique and can change over time, making impossible to predict the exact state at any future time t.
Nevertheless, given a starting joint distribution, it is possible to find the limiting joint distribution. Let us start by defining the iterative system for updating the joint distribution.

Definition 4.1.4 (Update Function of Joint Distribution)
Given a P.B.N. as in eq. (4.1.4), the initial joint probability distribution is denoted by $D(x_1, x_2, \ldots, x_n)$ over the n-dimensional hypercube, where $x_1, x_2, \ldots, x_n \in [0,\, 1]^n$. Then, the equation to compute the joint distribution of the network at the next iteration is:

$$D^{t+1} = \Psi(D^t), \tag{4.1.25}$$

where $\Psi : [0,\, 1]^{2^n} \to [0,\, 1]^{2^n}$. Then, the mapping Ψ is represented by the state transition matrix A so that the previous equation can be rewritten as:

$$D^{t+1} = D^t \cdot A = D^0 \cdot A^{t+1}, \tag{4.1.26}$$

where D^{t+1} and $D^t \in [0,\, 1]^{2^n}$, $D(0) = D(x)$ is the starting joint distribution and $A \in \mathbb{R}^{2^n \times 2^n}$ is the state transition matrix of Def. 4.1.3.

Assuming that the starting joint distribution is uniform, i.e. $D^0 = \left[\frac{1}{2^n},\, \frac{1}{2^n},\, \dots,\, \frac{1}{2^n}\right]$, it is possible to find the limiting probabilities of a P.B.N. iterating eq. (4.1.26). Let us see the procedure in the next example.

Example 4.1.5 (Limiting Probabilities)
Continuing Ex. 4.1.4, let us assume that the starting joint distribution is uniform, that is $D^0 = \left[\frac{1}{8},\, \frac{1}{8},\, \dots,\, \frac{1}{8}\right]$. Then, the limiting probabilities are found as follows:

$$\begin{aligned} D^{t+1} &= D^t \cdot A = \\ &= D^t \cdot D^{t-1} \cdot A = \\ &\vdots \\ &= D^0 \cdot A^{t+1}. \end{aligned} \tag{4.1.27}$$

After a sufficient number of iterations (around 57 iterations), one finally gets the limiting probabilities as:

$$\pi = [0,\, 0.828,\, 0,\, 0,\, 0,\, 0,\, 0.172,\, 0]. \tag{4.1.28}$$

This vector shows that, starting from a uniform distribution of the states, the active states will either be (110) or (001) in the long run. In some sense, the concept of limiting probabilities in P.B.N.s corresponds to the concept of attractors in B.N.s. However, unlike in a B.N. where the initial condition does not influence the attractors, the absorbing states in a P.B.N. might not be the same from a different starting distribution. If, for example, $D^0 = [1,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0]$ the limiting probabilities result:

$$\pi = [0,\, 1,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0], \tag{4.1.29}$$

which is different from the previous result since the active state will be (110) with probability one.

4.2 Graphical representation

Also Probabilistic Boolean Network nodes and their interactions are well represented by means of directed graphs. Let us define the network graph in this specific case.

Definition 4.2.1 (Network Graph of a Probabilistic Boolean Network)
Given a P.B.N., we associate with it a graph whose set of nodes is the set of boolean variables: $\mathbf{V} = \{\chi_1,\, \dots,\, \chi_n\}$. A pair $(\chi_i,\, \chi_j)$ is an edge belonging to $\mathbf{E}$ if $\chi_i(t)$ is an argument of $f_j^{\gamma_j}(\cdot)$ in eq. (4.1.1), thus the node $\chi_i(t)$ affects $\chi_j(t+1)$, with $0 < i,\, j \leq n$ and $0 < \gamma_j \leq l_j$. This directed graph is called network graph of the P.B.N.. Moreover, thanks to the equivalence between boolean variables and logic vectors, and consequently between the eqs. (4.1.1) and (4.1.3), the nodes of a network graph can either be $\chi_i \in \mathcal{B}$ or $x_i \in \mathcal{L}^2$, with $0 < i \leq n$ and $n \in \mathbb{N}^*$.

In the case of P.B.N., it is possible to define an incidence matrix for each subsystem Σ_γ as shown in the following.

Definition 4.2.2 (From Graph to Incidence Matrix)
Given a graph as in Def. 2.2.1, we define incidence matrix of subsystem Σ_γ the boolean matrix $\mathcal{I}(\mathbf{G},\, \Sigma_\gamma) \in \mathcal{B}^{n\times n}$ whose entries are defined based on the following rule:

$$[\mathcal{I}(\mathbf{G},\, \Sigma_\gamma)]_{i,\,j} = \begin{cases} 1, & \text{if } (V_i,\, V_j) \in \mathbf{E} \\ 0, & \text{otherwise} \end{cases}, \; \forall\, 0 < i,\, j \le n,\, 0 < \gamma \le N. \tag{4.2.1}$$

Algorithm 4.2.1 (From Incidence Matrix to Graph)
Given the incidence matrices $\mathcal{I}(\mathbf{G},\, \Sigma_\gamma) \in \mathcal{B}^{n\times n}$, $\forall\, 0 < \gamma \le N$ it is always possible to retrieve the corresponding graph.
Looking at each incidence matrix, one gets the edges connecting the $n \in \mathbb{N}^*$ nodes as follows:

$$[\mathcal{I}(\mathbf{G},\, \Sigma_\gamma)]_{i,\,j} = \begin{cases} 1 \to (V_i,\, V_j) \in \mathbf{E} \\ 0 \to (V_i,\, V_j) \notin \mathbf{E} \end{cases}, \; \forall\, 0 < i,\, j \le n,\, \forall\, 0 < \gamma \le N. \tag{4.2.2}$$

The next proposition shows an alternative way to derive the incidence matrix.

Proposition 4.2.1 (From Boolean Network to Incidence Matrix)
The incidence matrix $\mathcal{I}(\mathbf{G},\, \Sigma_\gamma) \in \mathcal{B}^{n\times n}$, $\forall\, 0 < \gamma \le N$ can be directly derived from the logic equations of a P.B.N as in eq. (4.1.3) in the following way:

$$[\mathcal{I}(\mathbf{G},\, \Sigma_\gamma)]_{i,\,j} = \begin{cases} 1, & \text{if } f_j^{\gamma_j}(\cdot) \text{ depends on } x_i(t) \\ 0, & \text{otherwise} \end{cases}, \; \forall\, 0 < i,\, j \le n,\, \forall\, 0 < \gamma \le N, \tag{4.2.3}$$

where the functions $f_j^{\gamma_j}$ are chosen accordingly with the selected subsystem Σ_γ.

Definition 4.2.3 (General Incidence Matrix)
Given the incidence matrices $\mathcal{I}(\mathbf{G},\, \Sigma_\gamma) \in \mathcal{B}^{n\times n}$, $\forall\, 0 < \gamma \le N$ it is possible to denote with $\mathcal{I}(\mathbf{G}) \in \mathcal{B}^{n\times n}$ the general incidence matrix of a P.B.N.. It can be found as follows:

$$\mathcal{I}(\mathbf{G}) = \bigvee_{\gamma=1}^{N} \mathcal{I}(\mathbf{G},\, \Sigma_\gamma). \tag{4.2.4}$$

An example is presented in the following.

Example 4.2.1 (Network Graph and Incidence Matrix)
Consider again Ex. 4.1.1. It is possible to build the network graph $\mathbf{G} \triangleq (\mathbf{V},\, \mathbf{E})$ applying Def. 2.2.3. The set of nodes is $\mathbf{V} = \{\chi_1,\, \chi_2,\, \chi_3\}$ and the set of edges is $\mathbf{E} = \{e_{1,1},\, e_{2,1},\, e_{1,2},\, e_{3,2},\, e_{2,3},\, e_{3,3}\}$. The incidence matrices can be retrieved either from Def. 4.2.2 or from Prop. 4.2.1 as follows:

$$\begin{aligned} \mathcal{I}(\mathbf{G},\, \Sigma_1) &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} & \mathcal{I}(\mathbf{G},\, \Sigma_2) &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \\ \mathcal{I}(\mathbf{G},\, \Sigma_3) &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} & \mathcal{I}(\mathbf{G},\, \Sigma_4) &= \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \end{aligned} \tag{4.2.5}$$

where the functions $f_j^{\gamma_j}$, with $0 < j \leq n$ and $0 < \gamma_j \leq N$, are associated to the subsystems Σ_γ according to matrix K of eq. (4.1.20). The resulting graph $\mathbf{G} \triangleq (\mathbf{V}, \mathbf{E})$ is reported in Fig. 4.2.1. The four incidence matrices are identical because, in this specific case, the functions $f_j^{\gamma_j}$ depend on the same nodes for $0 < \gamma_j \leq N$. Therefore, also the network graph is identical to that of Fig. 2.2.1 and the general incidence matrix is:

$$\mathcal{I}(\mathbf{G}) = \bigvee_{\gamma=1}^{4} \mathcal{I}(\mathbf{G},\, \Sigma_\gamma) = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}. \tag{4.2.6}$$

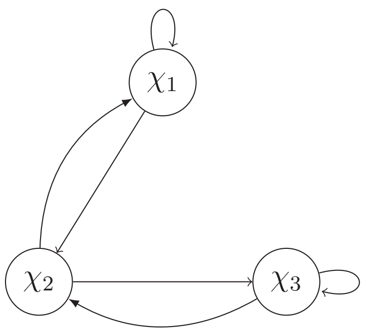

Figure 4.2.1: Network Graph of a Probabilistic Boolean Network

Since Rem. 2.2.1 can be applied also to probabilistic boolean networks, the state graph for P.B.N. is now presented.

Definition 4.2.4 (State Graph of a Probabilistic Boolean Network)
Consider a P.B.N. in algebraic form, described as in eq. (4.1.3). A directed graph $\mathcal{G} \triangleq \{\mathcal{V},\, \mathcal{E}\}$ is the state graph of the P.B.N. if its vertices correspond to all the values that the state vector can assume, i.e. $\mathcal{V} = \{\delta_{2^n}^i : i = 1, \ldots, 2^n\}$, and the edges $\in \mathcal{E}$ are the elements $e_{i,j} = \left(\delta_{2^n}^i,\, \delta_{2^n}^j\right)$ s.t. $[L_\gamma]_{j,i} = 1$, with $0 < i,\, j \leq 2^n$ and $0 < \gamma \leq N$ (equivalently, $\mathrm{Col}_i(L_\gamma) = \delta_{2^n}^j$).

Defs. 2.2.7, 2.2.9 and Prop. 2.2.2 are valid also for P.B.N. by substituting the deterministic structure matrix L with the probabilistic one L_γ. If in the state of a B.N. each node has one and only one outgoing edge (out-degree equal to one), in the state of a P.B.N. each node can have a number of outgoing edges equal to at most N, the number of subsystems Σ_γ. So the out-degree is $\leq N$. Instead, the in-degree of each node is $\leq 2^n \cdot N$ because there are exactly $2^n \cdot N$ edges.

Proposition 4.2.2 (Probabilistic Boolean Networks and State Graph)
Considering again the P.B.N. in eq. (4.1.4), it is possible to say that there exists a one-to-one correspondence between the P.B.N. and its state graph.

The following example shows how to build the state graph of a probabilistic boolean network.

Example 4.2.2 (State Graph)
Consider the P.B.N. of Ex. 4.1.2 which is fully characterized by the structure matrices L_γ for $\gamma = 1, 2, 3, 4$ of eq. (4.1.9). The set of vertices $\mathcal{V}$ and of edges $\mathcal{E}$ can

be found looking at the different L_γ. It follows that:

$$\begin{cases} \mathcal{V} = \left\{\delta_{2^3}^i : i = 1, 2, 3, 4, 5, 6, 7, 8\right\} \\ \mathcal{E} = \{(\delta_8^1, \delta_8^2), (\delta_8^1, \delta_8^4), (\delta_8^2, \delta_8^2), (\delta_8^3, \delta_8^1), \\ \qquad (\delta_8^3, \delta_8^3), (\delta_8^3, \delta_8^5), (\delta_8^3, \delta_8^7), (\delta_8^4, \delta_8^2), \\ \qquad (\delta_8^4, \delta_8^6), (\delta_8^5, \delta_8^4), (\delta_8^5, \delta_8^8), (\delta_8^6, \delta_8^2), \\ \qquad (\delta_8^6, \delta_8^4), (\delta_8^6, \delta_8^6), (\delta_8^6, \delta_8^8), (\delta_8^7, \delta_8^7), \\ \qquad (\delta_8^8, \delta_8^6), (\delta_8^8, \delta_8^8)\}. \end{cases} \tag{4.2.7}$$

Then, the state graph is reported in Fig. 4.2.2. Attached to each edge there is the probability that the edge is selected in the model Σ_γ. Equivalently, the probability indicates how likely it is for a state to move to a new state by means of that edge. For example, the nodes δ_8^2 and δ_8^7 are fixed points with probability one because, once they are reached, it is not possible to change state with any model Σ_γ. The edges colored in brown represent multiple edges from one state to another one and in fact the transition probability is a sum of probabilities.

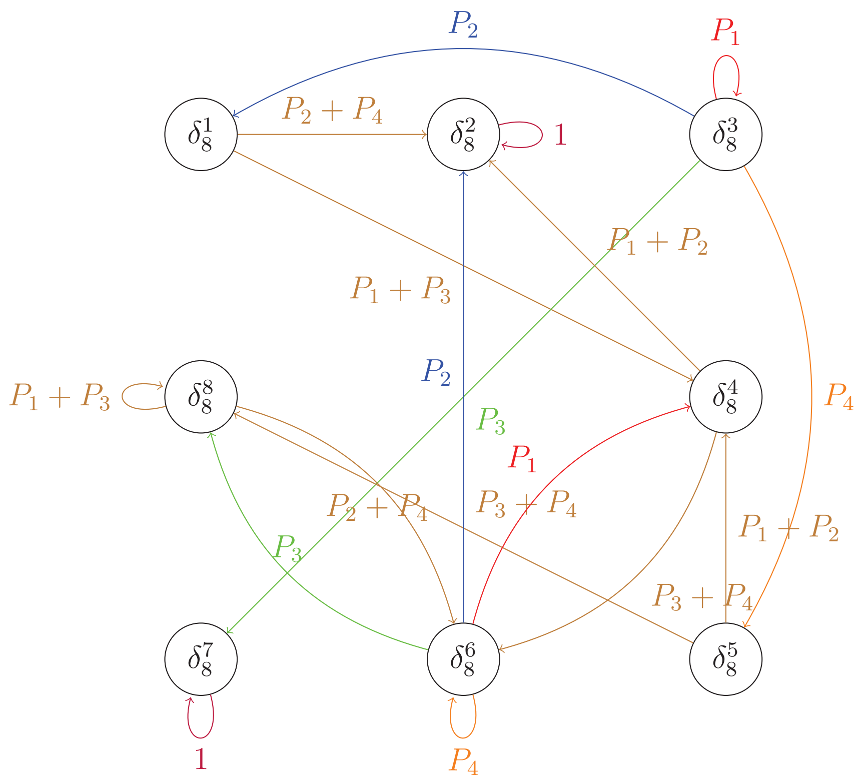

Figure 4.2.2: State Graph of a Probabilistic Boolean Network

4.3 Fixed Points and Limit Cycles

The concepts of fixed points and limit cycles play an important role in the study of stability of a P.B.N.. Differently to a B.N., a P.B.N. can be represented as a set of B.N.s, so for each node of the state graph there are N outgoing edges. This determines that fixed points and limit cycles are not deterministic but depend on

the probability that a specific model Σ_γ is active.
Let us start from the new definition of fixed point.

Definition 4.3.1 (Fixed Point)
Given a P.B.N. described as in eq. (4.1.4) and a time instant $t \in \mathbb{Z}$, a state $\delta_{2^n}^p$ is said to be a fixed point (or equilibrium point) of the P.B.N. if, for some $\gamma \in \{1, 2, \ldots, N\}$, it holds that:

$$\delta_{2^n}^p = \mathbf{x}(t+1) = L_\gamma \ltimes \mathbf{x}(t) = L_\gamma \ltimes \delta_{2^n}^p. \tag{4.3.1}$$

The previous definition shows the possibility that, for some models Σ_γ, the state $\delta_{2^n}^p$ is not actually a fixed point. That is why, in the case of a P.B.N., a stronger definition of fixed point is necessary.

Definition 4.3.2 (Common Fixed Point)
Given a P.B.N. described as in eq. (4.1.4) and a time instant $t \in \mathbb{Z}$, a state $\delta_{2^n}^p$ is said to be a common fixed point of the P.B.N. if, $\forall \gamma \in \{1, 2, \ldots, N\}$, it holds that:

$$\delta_{2^n}^p = \mathbf{x}(t+1) = L_\gamma \ltimes \mathbf{x}(t) = L_\gamma \ltimes \delta_{2^n}^p. \tag{4.3.2}$$

The following example clarifies the difference between the last two definitions.

Example 4.3.1 (Fixed Points and Common Fixed Points)
Consider the P.B.N. described by the structure matrices of eq. (4.1.9) whose state graph is shown in Fig. 4.2.2. Looking at the different L_γ, with $\gamma = 1, 2, \ldots, N$, one finds that the fixed points are the states δ_8^2, δ_8^3, δ_8^6, δ_8^7, δ_8^8 because there is at least one self-loop on each of these states. Instead, the common fixed points are a subset of the fixed points composed by the states δ_8^2 and δ_8^7, since all models have the same value in the corresponding structure matrix L_γ.

Definition 4.3.3 (Limit Cycle)
Given a P.B.N. described as in eq. (4.1.4), a limit cycle of length $t \in \mathbb{Z}$ is an ordered sequence of t distinct states $\{\delta_{2^n}^c, L_\gamma(0) \ltimes \delta_{2^n}^c, \ldots, L_\gamma(t-1) \ltimes \delta_{2^n}^c\}$, with $\gamma = 1, 2, \ldots, N$, if it holds that:

$$\begin{aligned} \delta_{2^n}^c = \mathbf{x}(t) &= L_\gamma(t-1) \ltimes L_\gamma(t-2) \ltimes \cdots \ltimes L_\gamma(0) \ltimes \mathbf{x}(0) = \\ &= L_\gamma(t-1) \ltimes L_\gamma(t-2) \ltimes \cdots \ltimes L_\gamma(0) \ltimes \delta_{2^n}^c. \end{aligned} \tag{4.3.3}$$

As in the case of a fixed point, each state can belong to more than one limit cycle due to the randomness in the choice of the structure matrix L_γ, which is visually represented by the multiple edges from each node in the state graph of a P.B.N..
Let us see an example about limit cycles.

Example 4.3.2 (Limit Cycles)
Consider the P.B.N. described by the structure matrices of eq. (4.1.9) whose state graph is shown in Fig. 4.2.2. From the latter, one can see that the limit cycles are $\{\delta_8^4, \delta_8^6\}$ and $\{\delta_8^6, \delta_8^8\}$. The first one requires that, when $\mathbf{x}(t) = \delta_8^4$, with $t \in \mathbb{Z}$, the active model is either Σ_3 or Σ_4 so that $\delta_8^6 = \mathbf{x}(t+1) = L_3 \ltimes \delta_8^4 = L_4 \ltimes \delta_8^4$, and, when $\mathbf{x}(t) = \delta_8^6$, the active model is Σ_1 so that $\delta_8^4 = \mathbf{x}(t+1) = L_1 \ltimes \delta_8^6$. The same reasoning can be followed for the second limit cycle.

The following proposition treats the convergence of a P.B.N. as opposed to the convergence of a deterministic and time invariant B.N..

Proposition 4.3.1 (Convergence of a Probabilistic Boolean Network)
Given a P.B.N. as in eq. (4.1.4), its dynamics may never converge to a specific attractor as in Def. 2.3.3. In detail, an initial state condition $\mathbf{x}(0) = \delta^i_{2^n}$, with $0 < i \leq 2^n$, can either converge to a fixed point or to a limit cycle, or evolve infinitely into different states of the network. Then, in general, it is no more possible to distinguish between attracting and transient states.

4.4 Stability of a Probabilistic Boolean Network

The property of stability, studied in Chapter 2 for boolean networks, is applicable also to P.B.N.s. Since the latter are probabilistic models, the stabilities that can be analysed are: with probability one, in probability and in distribution. In this thesis, only the first type of stability is treated. Its definition is presented in the following.

Definition 4.4.1 (Global Stability with probability one)
Consider a P.B.N. described as in eq. (4.1.3), where the probability P_γ, $\gamma = 1, 2, \ldots, N$, for the model Σ_γ to be active is as in eq. (4.1.6). Then, the P.B.N. is said to be globally stable with probability one if it globally converges to a fixed point with probability one. Specifically, there must exist a state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ such that, for any $\mathbf{x}_0 \in \mathcal{L}^{2^n}$:

$$\Pr\left\{\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}_e \,|\, \mathbf{x}(0) = \mathbf{x}_0\right\} = 1. \tag{4.4.1}$$

In order to present a theorem that contains an easily verifiable, necessary and sufficient condition, some preliminary lemmas are needed.

Lemma 4.4.1
Given a P.B.N. described as in eq. (4.1.3), with probability P_γ, $\gamma = 1, 2, \ldots, N$, for the model Σ_γ being active, denote the event:

$$A^T_\gamma = \{\Sigma_\gamma \text{ appears continuously over } T \text{ times}\}. \tag{4.4.2}$$

Then, for any $T > 0$, one gets:

$$\Pr\{A^T_\gamma \text{ happens infinite times}\} = 1. \tag{4.4.3}$$

This lemma shows that any network model can always appear in any continuous period. Thanks to it, the next lemma proves a necessary condition for global stability.

Lemma 4.4.2
Consider a P.B.N. described as in eq. (4.1.3), with probability P_γ, $\gamma = 1, \ldots, N$, for the model Σ_γ being active. If it globally converges to $\mathbf{x}_e \in \mathcal{L}^{2^n}$ with probability one, then $\mathbf{x}_e$ is a fixed point of any network model Σ_γ.

To provide the necessary and sufficient condition, some lemmas for Markov chains (see [23]) are needed.

Lemma 4.4.3
Consider a Markov chain ξ_n. A state j is said to be recurrent, if and only if:

$$\Pr\{\xi_n = j \text{ for infinitely many } n \,|\, \xi_0 = j\} = 1, \tag{4.4.4}$$

while a state j is said to be transient, if and only if:

$$\Pr\{\xi_n = j \text{ for infinitely many } n \,|\, \xi_0 = i\} = 1, \quad \forall\, i \neq j. \tag{4.4.5}$$

It is now possible to present the following theorem for stability.

Theorem 4.4.1 (Global Stability with probability one)
Consider a P.B.N. described as in eq. (4.1.3), with probability P_γ, $\gamma = 1, \ldots, N$, for the model Σ_γ being active. Let us denote:

$$L = \sum_{\gamma=1}^{N} L_\gamma. \tag{4.4.6}$$

Then, the P.B.N. globally converges to $\mathbf{x}_e = \delta_{2^n}^i \in \mathcal{L}^{2^n}$ with probability one, if and only if $\mathbf{x}_e$ is a fixed point of every model Σ_γ as in Def. 4.3.2, and

$$\text{Row}_i \left(\sum_{s=1}^{2^n} L^s \right) > 0. \tag{4.4.7}$$

Let us see an example.

Example 4.4.1 (Global Stability with probability one)
Consider a P.B.N. with two states, x_1 and $x_2 \in \mathcal{L}^2$, and its dynamics as follows:

$$\begin{cases} x_1(t+1) = f_1(x_1(t), x_2(t)) \\ x_2(t+1) = f_2(x_1(t), x_2(t)), \end{cases} \tag{4.4.8}$$

where $f_1 = \{f_1^1, f_1^2\}$ and $f_2 = \{f_2^1, f_2^2\}$ are chosen randomly with uniform distribution, so that there are $N = l_1 \cdot l_2 = 2 \cdot 2 = 4$ structure matrices:

$$\begin{aligned} L_1 &= \delta_4 \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}, \quad & L_2 &= \delta_4 \begin{bmatrix} 2 & 2 & 4 & 4 \end{bmatrix}, \\ L_3 &= \delta_4 \begin{bmatrix} 3 & 3 & 4 & 4 \end{bmatrix}, \quad & L_4 &= \delta_4 \begin{bmatrix} 4 & 3 & 4 & 4 \end{bmatrix}. \end{aligned} \tag{4.4.9}$$

It is easy to see that the unique common fixed point of all models Σ_γ, with $\gamma =$ 1, 2, 3, 4, is $\mathbf{x}_e = \delta_4^4$, as confirmed by the state graph of Fig. 4.4.1, where each edge has the transition probability attached. One can compute:

$$L = \sum_{\gamma=1}^{4} L_\gamma = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \tag{4.4.10}$$

where it is confirmed the presence of the common fixed point $\mathbf{x}_e = \delta_4^4$, and:

$$\sum_{s=1}^{4} L^s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{4.4.11}$$

Since $\text{Row}_4 \left(\sum_{s=1}^{4} L^s \right) > 0$, one can conclude that eq. (4.4.7) holds, so the P.B.N. is globally stable at $\mathbf{x}_e = \delta_4^4$.

Let us now see an example where there cannot be global stability with probability one.

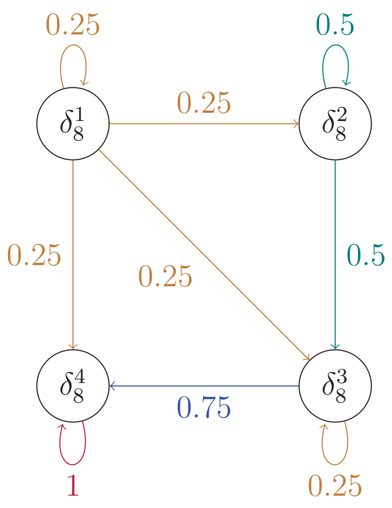

Figure 4.4.1: State Graph of a Probabilistic Boolean Network

Example 4.4.2 (Global Stability with probability one)
Refer again to Ex. 4.1.2 whose state graph is shown in Fig. 4.2.2. Consider that the update functions are chosen randomly with uniform distribution. Then, the structure matrix L is as follows:

$$L = \sum_{\gamma=1}^{4} L_\gamma = \begin{bmatrix} 0&0&1&0&0&0&0&0\\ 1&1&0&1&0&1&0&0\\ 0&0&1&0&0&0&0&0\\ 1&0&0&0&1&1&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&1&0&1\\ 0&0&1&0&0&0&1&0\\ 0&0&0&0&1&1&0&1 \end{bmatrix}, \tag{4.4.12}$$

where it is possible to recognize two common fixed points of all models, δ_8^2 and δ_8^7. Then, the boolean sum of the boolean powers of L is as follows:

$$\sum_{s=1}^{8} L^s = \begin{bmatrix} 0&0&1&0&0&0&0&0\\ 1&1&1&1&1&1&0&1\\ 0&0&1&0&0&0&0&0\\ 1&0&1&1&1&1&0&1\\ 0&0&1&0&0&0&0&0\\ 1&0&1&1&1&1&0&1\\ 0&0&1&0&0&0&1&0\\ 1&0&1&1&1&1&0&1 \end{bmatrix}. \tag{4.4.13}$$

In this case, there is no index $i \in \{1, 2, \ldots, 8\}$ such that $\mathrm{Row}_i\left(\sum_{s=1}^{8} L^s\right) > 0$. Then, the P.B.N. is not globally stable with probability one. This is due to the fact that, if $\mathbf{x}_0 = \delta_8^7$, the trajectory of the P.B.N. will never leave the fixed point making it impossible to reach the other common fixed point δ_8^2.

The references examined to write this chapter are the following: [6], [13], [14], [15], [24], [25], [26], [27], [28], [29], [30].

Chapter 5

Probabilistic Boolean Control Networks

The next step is to study the effects of inputs on P.B.N.s. In this case, a P.B.N. becomes a Probabilistic Boolean Control Network (P.B.C.N.). The properties of B.C.N.s studied in Chapter 3 are applicable to this new model.

5.1 Probabilistic Boolean Control Networks Dynamics

The structure of a probabilistic boolean control network is defined in the following.

Definition 5.1.1 (Probabilistic Boolean Control Network)
A P.B.C.N. is a discrete time dynamical system involving $n \in \mathbb{N}^*$ boolean state variables $\chi_i(t) \in \mathcal{B}$ and $m \in \mathbb{N}^*$ boolean inputs $\mu_j \in \mathcal{B}$, with $0 < i \leq n$ and $0 < j \leq m$. The evolution of the network is determined by a set of n logic first order difference equations:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t),\, \dots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \dots,\, \mu_m(t)\right) \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_2(t),\, \dots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \dots,\, \mu_m(t)\right) \\ \vdots \\ \chi_n(t+1) = f_n\left(\chi_1(t),\, \chi_2(t),\, \dots,\, \chi_n(t);\, \mu_1(t),\, \mu_2(t),\, \dots,\, \mu_m(t)\right), \end{cases} \tag{5.1.1}$$

where, at every time instant, each logic function f_i is selected from a collection of $l_i < \infty$ possible models, namely $f_i \in \{f_i^1,\, f_i^2,\, \dots,\, f_i^{l_i}\}$, with $f_i^{\gamma_i}(\cdot) : \mathcal{B}^n \times \mathcal{B}^m \to \mathcal{B}$, according to the probability of f_i being $f_i^{\gamma_i}$, i.e. $\Pr\{f_i = f_i^{\gamma_i}\} = p_i^{\gamma_i}$, with $\sum_{\gamma_i=1}^{l_i} p_i^{\gamma_i} = 1$, $\gamma_i \in \{1,\, 2,\, \dots,\, l_i\}$ and $i \in \{1,\, 2,\, \dots,\, n\}$.

A P.B.C.N. can be rewritten in its algebraic form by means of the same method used for P.B.N.s. Then, one needs to convert all boolean state variables, inputs and functions into their algebraic representation. This is explained in the following proposition.

Proposition 5.1.1 (Probabilistic Boolean Control Network in Algebraic Form)
A P.B.C.N. as in eq. (5.1.1) can be transformed into its equivalent algebraic form following Thm. 1.4.3:

$$\mathbf{x}(t+1) = L_{\gamma(t)} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \tag{5.1.2}$$

where $\mathbf{x}(\cdot) \in \mathcal{L}^{2^n}$ is the state vector, $\mathbf{u}(\cdot) \in \mathcal{L}^{2^m}$ is the input vector and $L_{\gamma(t)} \in \mathcal{L}^{2^n \times 2^{n+m}}$ is one of the N transition matrices of the P.B.C.N. as in eq. (4.1.4) for P.B.N.s. To simplify the notation, if the active subsystem at time t, $\gamma(t)$, is called $\bar{\gamma}$, then the transition matrix $L_{\gamma(t)}$ becomes $L_{\bar{\gamma}}$. Then, the previous equation can be rewritten as follows:

$$\mathbf{x}(t+1) = L_{\bar{\gamma}} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \tag{5.1.3}$$

Using Thm. 1.4.2, one can obtain the following system:

$$\begin{cases} x_1(t+1) & = f_1(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) & = f_2(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ \vdots & \\ x_n(t+1) & = f_n(x_1(t),\, x_2(t),\, \dots,\, x_n(t);\, u_1(t),\, u_2(t),\, \dots,\, u_m(t)) = \\ & = M_{f_n} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \end{cases} \tag{5.1.4}$$

where $x_i(\cdot) \in \mathcal{L}^2$, $\forall\, i$, $0 < i \le n$, $u_j(\cdot) \in \mathcal{L}^2$, $\forall\, j$, $0 < j \le m$, $M_{f_i} \in \mathcal{L}^{2^n \times 2^{n+m}}$, with $M_{f_i} \in \{M_{f_i}^1,\, M_{f_i}^2,\, \dots,\, M_{f_i}^{l_j}\}$, and $\mathbf{x}(t) = \ltimes_{i=1}^n x_i(t) \in \mathcal{L}^{2^n}$, $\mathbf{u}(t) = \ltimes_{j=1}^m u_j(t) \in \mathcal{L}^{2^m}$.

In eq. (5.1.2), it is customary to put the input vector $\mathbf{u}(\cdot)$ before the state vector $\mathbf{x}(\cdot)$. Let us remark that this choice does not alter the dynamics of the P.B.C.N. since the two vectors can be exchanged by means of a proper swap matrix.

Example 5.1.1 (Probabilistic Boolean Control Network in Algebraic Form)
Given a P.B.C.N. with $n = 3$ state variables, $m = 2$ inputs and the function sets f_1, f_2 and $f_3 : \mathcal{B}^3 \to \mathcal{B}$, where $f_1 = \{f_1^1,\, f_1^2\}$, $f_2 = \{f_2^1,\, f_2^2\}$ and $f_3 = \{f_3^1\}$ as follows:

$$\begin{cases} \chi_1(t+1) = f_1\left(\chi_1(t),\, \chi_2(t),\, \mu_1(t)\right) = \\ \quad = \begin{cases} f_1^1\left(\chi_1(t),\, \chi_2(t),\, \mu_1(t)\right) = (\chi_1(t) \wedge \chi_2(t)) \vee \mu_1(t), & p_1^1 = \Pr\{f_1 = f_1^1\} = 0.6 \\ f_1^2\left(\chi_1(t),\, \chi_2(t),\, \mu_1(t)\right) = (\chi_1(t) \vee \chi_2(t)) \vee \mu_1(t), & p_1^2 = \Pr\{f_1 = f_1^2\} = 0.4 \end{cases} \\ \chi_2(t+1) = f_2\left(\chi_1(t),\, \chi_3(t),\, \mu_2(t)\right) = \\ \quad = \begin{cases} f_2^1\left(\chi_1(t),\, \chi_3(t),\, \mu_2(t)\right) = (\chi_2(t) \vee \chi_3(t)) \wedge \mu_2(t), & p_2^1 = \Pr\{f_2 = f_2^1\} = 0.7 \\ f_2^2\left(\chi_1(t),\, \chi_3(t),\, \mu_2(t)\right) = (\chi_2(t) \wedge \chi_3(t)) \wedge \mu_2(t), & p_2^2 = \Pr\{f_2 = f_2^2\} = 0.3 \end{cases} \\ \chi_3(t+1) = f_3\left(\chi_2(t),\, \chi_3(t)\right) = \\ \quad = f_3^1\left(\chi_2(t),\, \chi_3(t)\right) = \neg\chi_2(t) \wedge \chi_3(t), \quad p_3^1 = \Pr\{f_3 = f_3^1\} = 1, \end{cases} \tag{5.1.5}$$

it is possible to rewrite the system as:

$$\begin{cases} x_1(t+1) = M_{f_1} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_2(t+1) = M_{f_2} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t) \\ x_3(t+1) = M_{f_3} \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \end{cases} \tag{5.1.6}$$

where $M_{f_i} \in \{M_{f_i}^1,\, M_{f_i}^2,\, \dots,\, M_{f_i}^{l_i}\} \in \mathcal{L}^{2 \times 32}$, with $1 \le i \le 3$. Finally, the $N = l_1 \cdot l_2 \cdot l_3 = 2 \cdot 2 \cdot 1 = 4$ structure matrices in the P.B.C.N. description (5.1.2) are found by means of the Khatri-Rao product:

$$\begin{aligned}
&L_1 = M^1_{f_1} * M^1_{f_2} * M^1_{f_3} = \\
&= \delta_8 \begin{bmatrix} 2 & 2 & 2 & 4 & 1 & 2 & 1 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 3 & 4 & 2 & 2 & 6 & 8 & 5 & 6 & 5 & 8 & 4 & 4 & 8 & 8 & 7 & 8 & 7 & 8 \end{bmatrix}, \\
&\text{with } P_1 = \Pr\{\Sigma_1 \text{ selected}\} = p^1_1 \cdot p^1_2 \cdot p^1_3 = 0.42; \\
&L_2 = M^1_{f_1} * M^2_{f_2} * M^1_{f_3} = \\
&= \delta_8 \begin{bmatrix} 2 & 4 & 4 & 4 & 1 & 4 & 3 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 3 & 4 & 2 & 4 & 8 & 8 & 5 & 8 & 7 & 8 & 4 & 4 & 8 & 8 & 7 & 8 & 7 & 8 \end{bmatrix}, \\
&\text{with } P_2 = \Pr\{\Sigma_2 \text{ selected}\} = p^1_1 \cdot p^2_2 \cdot p^1_3 = 0.18; \\
&L_3 = M^2_{f_1} * M^1_{f_2} * M^1_{f_3} = \\
&= \delta_8 \begin{bmatrix} 2 & 2 & 6 & 8 & 5 & 6 & 5 & 8 & 4 & 4 & 8 & 8 & 7 & 8 & 7 & 8 & 6 & 6 & 6 & 8 & 5 & 6 & 5 & 8 & 8 & 8 & 8 & 8 & 7 & 8 & 7 & 8 \end{bmatrix}, \\
&\text{with } P_3 = \Pr\{\Sigma_3 \text{ selected}\} = p^2_1 \cdot p^1_2 \cdot p^1_3 = 0.28; \\
&L_4 = M^2_{f_1} * M^2_{f_2} * M^1_{f_3} = \\
&= \delta_8 \begin{bmatrix} 2 & 4 & 4 & 4 & 1 & 4 & 3 & 4 & 4 & 4 & 4 & 4 & 3 & 4 & 3 & 4 & 2 & 4 & 4 & 4 & 1 & 4 & 7 & 8 & 4 & 4 & 4 & 4 & 3 & 4 & 7 & 8 \end{bmatrix}, \\
&\text{with } P_4 = \Pr\{\Sigma_4 \text{ selected}\} = p^2_1 \cdot p^2_2 \cdot p^1_3 = 0.12.
\end{aligned} \tag{5.1.7}$$

where $L_i \in \mathcal{L}^{8\times 32}$, with $i = 1, 2, 3, 4$.

5.1.1 Probabilistic Boolean Control Networks as Switched Systems

In the previous chapters it has been shown that, in a B.C.N., the selection of a specific boolean network depends on the deterministic value of the inputs, while, in a P.B.N., the same selection depends on the probability that a specific model Σ_γ is selected at time $t \in \mathbb{Z}$. In a P.B.C.N., which is a mixture of a B.C.N. and a P.B.N., the choice depends on both the deterministic value of the inputs and on the probability that a specific model is selected. Algebraically, it holds:

$$\mathbf{x}(t+1) = L_\gamma(t) \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \tag{5.1.8}$$

where $L_\gamma(t) \in \mathcal{L}^{2^n \times 2^{n+m}}$ is the structure matrix of the active B.C.N. at time t. L_γ is selected by the probability $P_\gamma = \Pr\{\Sigma_\gamma \text{ selected}\}$, with $0 < \gamma \leq N$. Therefore, the P.B.C.N. can be written as a switched system:

$$\mathbf{x}(t+1) = L \ltimes \gamma(t) \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \tag{5.1.9}$$

where $\gamma(t) \in \mathcal{L}^N$ selects with $\Pr\{\gamma(t) = \delta^\gamma_N\} = P_\gamma$ the active structure matrix L_γ taking values in $\{1, \ldots, N\}$, $\mathbf{u}(t) \in \mathcal{L}^{2^m}$ is the input selecting the sub-model of L_γ and L is as follows:

$$L = \begin{bmatrix} L_1 & L_2 & \ldots & L_N \end{bmatrix}. \tag{5.1.10}$$

Let us now consider an example.

Example 5.1.2 (Probabilistic Boolean Control Networks as Switched Systems)
Consider the P.B.C.N. of Ex. 5.1.1. Since there are two possible update functions for the state x_1, two for the state x_2 and one for the state x_3, i.e. $f_1 = \{f^1_1, f^2_1\}$, $f_2 = \{f^1_2, f^2_2\}$ and $f_3 = \{f^1_3\}$, this means that there are $N = l_1 \cdot l_2 \cdot l_3 = 4$ sub-networks.

Moreover, there are two inputs which means that the number of combinations is $2^m = 4$. Therefore, the matrix L is as follows:

$$L = \begin{bmatrix} L_1 & L_2 & L_3 & L_4 \end{bmatrix} \in \mathcal{L}^{8\times 128}, \tag{5.1.11}$$

and the vectors have dimensions $\gamma(t) \in \mathcal{L}^4$ and $\mathbf{u}(t) \in \mathcal{L}^4$.

5.1.2 Graphical Representation

Also in the case of a P.B.C.N., it is possible to represent graphically the network by means of a state graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as in Def. 2.2.6. The next definition introduces the network graph of a P.B.C.N..

Definition 5.1.2 (Network Graph of a Probabilistic Boolean Control Network)
Given a P.B.C.N., we associate with it a graph $(\mathbf{V}, \mathbf{E})$ whose set of nodes is the set of boolean variables: $\mathbf{V} = \{\chi_1, \ldots, \chi_n\}$. A pair (χ_i, χ_j) is an edge belonging to $\mathbf{E}$ if $\chi_i(t)$ is an argument of $f_j^{\gamma_j}(\cdot)$ in eqn. (5.1.1), thus the value of $\chi_i(t)$ affects the value $\chi_j(t+1)$, with $0 < i, j \leq n$ and $0 < \gamma_j \leq l_j$. The same holds for a pair including an input $\mu_k(t)$ with $0 < k \leq m$. (μ_k, χ_j) is an edge belonging to $\mathbf{E}$ if $\mu_k(t)$ is an argument of $f_j^{\gamma_j}(\cdot)$. This directed graph is called network graph of the P.B.C.N..

It is also possible to represent a P.B.C.N. by means of a state graph as expressed in the following definition.

Definition 5.1.3 (State Graph of a Probabilistic Boolean Control Network)
Consider a P.B.C.N. in algebraic form, described as in eqn. (5.1.4). The state graph of the P.B.C.N. is a directed graph $\mathcal{G} \triangleq \{\mathcal{V}, \mathcal{E}\}$ whose vertices correspond to all the values that the state vector can assume, i.e. $\mathcal{V} = \{\delta_{2^n}^i : i = 1, \ldots, 2^n\}$, and whose edges $\in \mathcal{E}$ are the elements $e_{i,j} = \left(\delta_{2^n}^i, \delta_{2^n}^j\right)$ s.t. $\left[[L_\gamma]_k\right]_{j,i} = 1$, with $0 < i, j \leq 2^n$, $0 < k \leq 2^m$ and $0 < \gamma \leq N$.

In order to draw the state graph of a P.B.C.N., it is possible to introduce the matrix L_{TOT} that summarizes the information brought by the single structure matrices L_γ, with $\gamma = 1, 2, 3, 4$. In fact, the information of each of the N B.C.N.s is collected in the matrix $L_{tot,\gamma}$ of eqn. (3.1.18). Then, L_{TOT} is as follows:

$$L_{TOT} = \sum_{\gamma=1}^{N} L_{tot,\gamma}. \tag{5.1.12}$$

The following example shows this procedure.

Example 5.1.3 (Graphical Representation)
Given a P.B.C.N. in algebraic form with two state variables and one input belonging to $\mathcal{L}^2$, and two possibilities for each state update function so that there are $N = 4$ values of L_γ with uniform distribution, with $\gamma = 1, 2, 3, 4$, the state update equation is:

$$\mathbf{x}(t+1) = L_\gamma \ltimes \mathbf{u}(t) \ltimes \mathbf{x}(t), \tag{5.1.13}$$

where $\mathbf{x}(\cdot) \in \mathcal{L}^4$, $\mathbf{u}(\cdot) \in \mathcal{L}^2$ and the different $L_\gamma \in \mathcal{L}^{4\times 8}$ have the following entries:

$$\begin{aligned} L_1 &= \delta_4 \begin{bmatrix} 4 & 3 & 4 & 4 & 1 & 4 & 4 & 4 \end{bmatrix} \\ L_2 &= \delta_4 \begin{bmatrix} 2 & 3 & 1 & 4 & 1 & 4 & 2 & 4 \end{bmatrix} \\ L_3 &= \delta_4 \begin{bmatrix} 1 & 2 & 2 & 4 & 4 & 3 & 2 & 4 \end{bmatrix} \\ L_4 &= \delta_4 \begin{bmatrix} 4 & 3 & 1 & 4 & 3 & 1 & 2 & 4 \end{bmatrix}. \end{aligned} \tag{5.1.14}$$

By considering that each L_γ represents a single B.C.N., one can compute the matrix $L_{tot,\gamma}$ using eqn. (3.1.18) for each L_γ and then superimpose the state graphs corresponding to each matrix $L_{tot,\gamma}$.
The matrices $L_{tot,\gamma}$, for $\gamma = 1,\, 2,\, 3,\, 4$ are:

$$\begin{aligned} L_{tot,1} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad L_{tot,2} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \\ L_{tot,3} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad L_{tot,4} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{5.1.15}$$

Superimposing the state graphs corresponds to the logical sum of the matrices $L_{tot,\gamma}$, denoted by L_{TOT} (see eqn. (5.1.12)) and with entries:

$$L_{TOT} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.1.16}$$

Thanks to the information embedded in L_{TOT}, one can build the corresponding state graph $\mathcal{G}$ where the common nodes and edges are:

$$\begin{cases} \mathcal{V} = \{\delta_4^1,\, \delta_4^2,\, \delta_4^3,\, \delta_4^4\} \\ \mathcal{E} = \{(\delta_4^1,\, \delta_4^1),\, (\delta_4^1,\, \delta_4^2),\, (\delta_4^1,\, \delta_4^3),\, (\delta_4^1,\, \delta_4^4), \\ \qquad (\delta_4^2,\, \delta_4^1),\, (\delta_4^2,\, \delta_4^2),\, (\delta_4^2,\, \delta_4^3),\, (\delta_4^2,\, \delta_4^4), \\ \qquad (\delta_4^3,\, \delta_4^1),\, (\delta_4^3,\, \delta_4^2),\, (\delta_4^3,\, \delta_4^4), \\ \qquad (\delta_4^4,\, \delta_4^4)\}. \end{cases} \tag{5.1.17}$$

Finally, the state graph is reported in Fig. 5.1.1.

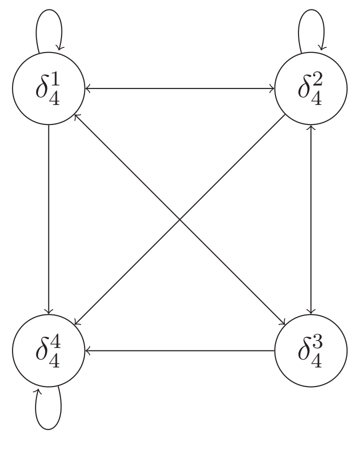

Figure 5.1.1: State Graph of a Probabilistic Boolean Control Network

5.2 Reachability

As done for the deterministic B.C.N., the analysis of reachability for a P.B.C.N. is now proposed. Let us start by introducing a new concept, called reachability with probability one.

Definition 5.2.1 (Reachability from a State)
Given a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, a state $\mathbf{x}_j \in \mathcal{L}^{2^n}$ is said to be reachable from the state $\mathbf{x}_i \in \mathcal{L}^{2^n}$, if there exists a sequence of model-control pairs $\{(\Sigma(t), u(t)), t = 0, 1, \ldots, s-1\}$, $s < \infty$, such that the controlled network trajectory will reach $\mathbf{x}_j$ at time s.

The following two more general definitions follow from the previous definition.

Definition 5.2.2 (Reachability)
By referring to Def. 5.2.1, a state $\mathbf{x}_j \in \mathcal{L}^{2^n}$ is said to be reachable if it is reachable from any state $\mathbf{x}_i \in \mathcal{L}^{2^n}$.

Definition 5.2.3 (Reachability of P.B.C.N.)
By referring to Def. 5.2.1, a P.B.C.N. is said to be reachable if any $\mathbf{x}_j \in \mathcal{L}^{2^n}$ is reachable.

It is now possible to explore an equivalent computable condition for reachability. Given that L_γ denotes the structure matrix of the γ-th network model Σ_γ, one can compute the following two matrices:

$$M_\gamma = \sum_{j=1}^{2^m} \mathrm{Blk}_j(L_\gamma), \tag{5.2.1}$$

and

$$M_P = \sum_{\gamma=1}^{N} M_\gamma. \tag{5.2.2}$$

$[M_P]_{i,j} = 1$, i.e. the i, j-th entry of matrix M_P is equal to one, means that there exists at least one model Σ_γ and a control u such that the state $\delta_{2^n}^i$ can be reached from the state $\delta_{2^n}^j$ in one step.
Now, one can define the reachability matrix for P.B.C.N.s, as done in the following definition.

Definition 5.2.4 (Reachability Matrix)
Given a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, the reachability matrix is a boolean matrix belonging to $\mathcal{B}^{2^n \times 2^n}$ defined as follows:

$$\mathcal{R}_P = \sum_{s=1}^{2^n} M_P^s. \tag{5.2.3}$$

The new matrix $\mathcal{R}_P$ can completely characterize the reachability as shown in the following theorem.

Theorem 5.2.1 (Reachability and $\mathcal{R}_P$)
A P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, is reachable if and only if $\mathcal{R}_P > 0$.

The following example clarifies the concept of reachability for a P.B.C.N..

Example 5.2.1 (Reachability)
Consider Ex. 5.1.1. The matrices M_γ, $\gamma = 1, 2, 3, 4$, are found using eqn. (5.2.1):

$$\begin{aligned}
M_1 &= \begin{bmatrix} 0&0&0&0&1&0&1&0\\ 1&1&1&0&0&1&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1\\ 0&0&0&0&1&0&1&0\\ 0&0&1&0&0&1&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&1&1&0&1&0&1 \end{bmatrix}, \quad
M_2 = \begin{bmatrix} 0&0&0&0&1&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 0&0&1&1&0&1&0&1 \end{bmatrix}, \\
M_3 &= \begin{bmatrix} 0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&0&0&1&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1 \end{bmatrix}, \quad
M_4 = \begin{bmatrix} 0&0&0&0&1&0&0&0\\ 1&0&0&0&0&0&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{bmatrix}.
\end{aligned} \tag{5.2.4}$$

Then, matrix M_P is found using eqn. (5.2.2):

$$M_P = \begin{bmatrix} 0&0&0&0&1&0&1&0\\ 1&1&1&0&0&1&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1\\ 0&0&0&0&1&0&1&0\\ 1&1&1&0&0&1&0&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&0&1&0&1 \end{bmatrix}. \tag{5.2.5}$$

It is now possible to compute the reachability matrix using eqn. (5.2.3):

$$\mathcal{R}_P = \begin{bmatrix} 0&0&0&0&1&0&1&0\\ 1&1&1&0&1&1&1&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&1&1&1&1\\ 0&0&0&0&1&0&1&0\\ 1&1&1&0&1&1&1&0\\ 0&0&0&0&1&0&1&0\\ 1&1&1&1&1&1&1&1 \end{bmatrix}. \tag{5.2.6}$$

Since there are some entries equal to zero, one can conclude that the P.B.C.N. is not reachable. For example, $[\mathcal{R}_P]_{3,2} = 0$ indicates that state δ_8^3 cannot be reached from state δ_8^2. Nevertheless, some conclusions about the reachability of specific states can be drawn. The fourth and eighth rows contain only ones, so the states δ_8^4 and δ_8^8 are reachable because they can be reached from any state in the network.

5.3 Controllability

Let us now analyse the controllability property for a P.B.C.N..

Definition 5.3.1 (Controllability with Probability One to a State)
Given a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, a state $\mathbf{x}_i \in \mathcal{L}^{2^n}$ is said to be controllable with probability one to the state $\mathbf{x}_j \in \mathcal{L}^{2^n}$ if there exists a control sequence such that:

$$\Pr\{\mathbf{x}(t) = \mathbf{x}_j \text{ for some } t \geq 1 \,|\, \mathbf{x}(0) = \mathbf{x}_i\} = 1. \tag{5.3.1}$$

The following two more general definitions follow from the previous definition.

Definition 5.3.2 (Controllability with Probability One)
By referring to Def. 5.3.1, a P.B.C.N. is said to be controllable with probability one at state $\mathbf{x}_i$ if it is controllable with probability one from $\mathbf{x}_i$ to any $\mathbf{x}_j \in \mathcal{L}^{2^n}$.

Definition 5.3.3 (Controllability with Probability One of a P.B.C.N.)
By referring to Def. 5.3.1, a P.B.C.N. is said to be controllable with probability one if it is controllable at any $\mathbf{x}_i \in \mathcal{L}^{2^n}$.

It is now possible to characterise the controllability by means of the reachability matrix thanks to the following theorem.

Theorem 5.3.1 (Controllability with Probability One and $\mathcal{R}_P$)
Consider a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$. It is controllable with probability one if and only if $\mathcal{R}_P > 0$.

Let us now see an example where the controllability property is investigated.

Example 5.3.1 (Controllability with Probability One)
Consider Exs. 5.1.1 and 5.2.1. By looking at matrix $\mathcal{R}_P$ of eqn. (5.2.6) and observing that some entries are equal to zero, one can conclude that the P.B.C.N. is not controllable with probability one. Nevertheless, the fifth and seventh columns contain only ones which means that the states δ_8^5 and δ_8^7 are controllable with probability one to any state $\in \mathcal{L}^{2^n}$.

5.4 Stabilisation

In this section, the stabilisation of a P.B.C.N. is explored.
The concept of control-fixed point is now introduced because it will be frequently used in the following.

Definition 5.4.1 (Control-fixed Point)
In a P.B.C.N. as in eqn. (5.1.1), a point $\mathbf{x}_e \in \mathcal{L}^{2^n}$ is called a control-fixed point if there exists a control $\mathbf{u}_e \in \mathcal{L}^{2^m}$ such that $\mathbf{x}_e = L_{\bar{\gamma}} \ltimes \mathbf{u}_e \ltimes \mathbf{x}_e$.

One can check the control-fixed points of a network directly from the structure matrix L as explained in the following proposition.

Proposition 5.4.1 (Control-fixed Point)
The state $\mathbf{x}_e = \delta_{2^n}^i \in \mathcal{L}^{2^n}$ is a control-fixed point under the control $\mathbf{u}_e = \delta_{2^m}^j \in \mathcal{L}^{2^m}$ if and only if $\mathbf{x}_e = \mathrm{Col}_i(\mathrm{Blk}_j(L_{\bar{\gamma}}))$.

It is now possible to define the stabilisation with probability one of a P.B.C.N..

Definition 5.4.2 (Stabilisation with Probability One)
Consider a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$. It is said to be stabilisable with probability one if there exists a control sequence such that the controlled system converges to a state $\mathbf{x}_e \in \mathcal{L}^{2^n}$ with probability one.

From the concept of stability of P.B.N.s, it is possible to suppose that the necessary condition of stabilisation of a P.B.C.N. is the fact that the state $\mathbf{x}_e$ is a control-fixed point of each network model. This is explained in the next proposition.

Proposition 5.4.2 (Stabilisation with Probability One)
Given a P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, a necessary condition for the system to be stabilisable with probability one to $\mathbf{x}_e = \delta^i_{2^n} \in \mathcal{L}^{2^n}$ is that $\mathbf{x}_e$ is a common control-fixed point of all network models Σ_γ under the same control $\mathbf{u}_e = \delta^j_{2^m} \in \mathcal{L}^{2^m}$.

The sufficient condition is given in the following theorem.

Theorem 5.4.1 (Stabilisation with Probability One)
A P.B.C.N. as in eq. (5.1.1), with N network models Σ_γ, $\gamma = 1, 2, \ldots, N$, is stabilisable to $\mathbf{x}_e = \delta^i_{2^n} \in \mathcal{L}^{2^n}$ with probability one if and only if the necessary condition in Prop. 5.4.2 holds and:

$$\mathrm{Row}_i(\mathcal{R}_\mathcal{P}) > 0. \tag{5.4.1}$$

Let us consider an example in this regard.

Example 5.4.1 (Stabilisation with Probability One)
Consider the setup of Ex. 5.3.1. By observing the entries of matrix $\mathcal{R}_P$, since $\mathrm{Row}_4(\mathcal{R}_\mathcal{P})$ and $\mathrm{Row}_8(\mathcal{R}_\mathcal{P}) > 0$, one can conclude that the P.B.C.N. is stabilisable to the states δ^4_8 and δ^8_8.

5.5 Feedback Control

The previous section shows which states a P.B.C.N. is stabilisable to with probability one. This section proposes a solution to guarantee that every state of the network converges to a control-fixed point. Similarly to what has been done for B.C.N.s, a state feedback controller is designed for P.B.C.N.s. Let us start by defining the problem.
Given a P.B.C.N. as in eqn. (5.1.1), the problem of stabilisation of the network is solved using a time-invariant feedback law, where the input vector is a linear combination of state vector entries:

$$\mathbf{u}(t) = K \ltimes \mathbf{x}(t), \tag{5.5.1}$$

where $K \in \mathcal{L}^{2^m \times 2^n}$ is the feedback matrix. The result is that the overall closed-loop system:

$$\mathbf{x}(t+1) = L_{in} \ltimes K \ltimes \mathbf{x}(t) \ltimes \mathbf{x}(t), \tag{5.5.2}$$

where $\mathbf{x}(t) \in \mathcal{L}^{2^n}$, $\mathbf{u}(t) \in \mathcal{L}^{2^m}$ and $L_{in} \in \mathcal{L}^{2^n \times 2^{n+m}}$ is defined as follows:

$$L_{in} = \sum_{\gamma=1}^{N} L_\gamma, \tag{5.5.3}$$

is globally stable with probability one.
Let us start by defining a sequence of sets.

Definition 5.5.1 (Recursive Sets)
Consider a P.B.C.N. as in eqn. (5.1.1). If $\mathbf{x}' \in \mathcal{L}^{2^n}$ is the fixed point to which the network needs to be stabilised, the sets $\{\Omega_k(\mathbf{x}')\}$ are defined recursively as follows:

$$\begin{cases} \Omega_1(\mathbf{x}') = \{a \in \mathcal{L}^{2^n} : \text{there is a}\, \mathbf{u} \in \mathcal{L}^{2^m} \,\text{such that} \\ \qquad\qquad \Pr\{\mathbf{x}(t+1) = \mathbf{x}' \,|\, \mathbf{x}(t) = a,\, \mathbf{u}(t) = \mathbf{u}) = 1\}, \\ \Omega_{k+1}(\mathbf{x}') = \{a \in \mathcal{L}^{2^n} : \text{there is a}\, \mathbf{u} \in \mathcal{L}^{2^m} \,\text{such that the conditions} \\ \qquad\qquad \Pr\{\mathbf{x}(t+1) = b \,|\, \mathbf{x}(t) = a,\, \mathbf{u}(t) = \mathbf{u}) > 0,\, b \in \mathcal{L}^{2^n} \\ \qquad\qquad \text{imply that}\, b \in \{\Omega_k(\mathbf{x}')\},\, k = 1,\, 2,\, 3,\, \ldots. \end{cases} \tag{5.5.4}$$

The two next lemmas give some basic properties of the sets $\Omega_k(\mathbf{x}')$.

Lemma 5.5.1
If $\mathbf{x}' \in \Omega_1(\mathbf{x}')$, then $\Omega_k(\mathbf{x}') \subseteq \Omega_{k+1}(\mathbf{x}')$ for $k \geq 1$.

Lemma 5.5.2
The following two items are true:

- If $\Omega_1(\mathbf{x}') = \{\mathbf{x}'\}$, then $\Omega_k(\mathbf{x}') = \{\mathbf{x}'\}$ for all $k \geq 1$;
- If $\Omega_{j+1}(\mathbf{x}') = \Omega_j(\mathbf{x}')$ for some $j \geq 1$, then $\Omega_k(\mathbf{x}') = \Omega_j(\mathbf{x}')$ for all $k \geq j$.

The following theorem can now be presented.

Theorem 5.5.1 (State Feedback and Recursive Sets)
Consider a P.B.C.N. as in eqn. (5.1.1) and let $\mathbf{x}' = x_1' \ltimes x_2' \ltimes \cdots \ltimes x_n'$. If there is a state feedback law $\mathbf{u}(\cdot)$ such that $\mathbf{x}'$ is globally stable with probability one for the closed-loop system of eqn. (5.5.2), then it holds that:

- $\mathbf{x}' \in \Omega_1(\mathbf{x}')$;
- there exists an integer $G \leq 2^n - 1$ such that $\Omega_G(\mathbf{x}') = \mathcal{L}^{2^n}$.

The previous conditions in Thm. 5.5.1 are also sufficient for the existence of a state feedback controller that globally stabilises the P.B.C.N.. This is shown in the following theorem, after the introduction of some notation.
With the conditions in Thm. 5.5.1 satisfied, $\mathcal{L}^{2^n}$ can be rewritten as the union of the disjoints sets previously defined:

$$\mathcal{L}^{2^n} = \Omega_1(\mathbf{x}') \cup (\Omega_2(\mathbf{x}') \setminus \Omega_1(\mathbf{x}')) \cup \cdots \cup (\Omega_G(\mathbf{x}') \setminus \Omega_{G-1}(\mathbf{x}')). \tag{5.5.5}$$

Then, to each $1 \leq i \leq 2^n$ there corresponds a unique integer $1 \leq k_i \leq G$ such that $\delta_{2^n}^i \in \Omega_{k_i}(\mathbf{x}') \setminus \Omega_{k_i-1}(\mathbf{x}')$, where $\Omega_0(\mathbf{x}') = \emptyset$.
If $k_i = 1$, then one can choose a vector $v_i \in \mathcal{L}^{2^m}$ such that:

$$\Pr\{\mathbf{x}(t+1) = \mathbf{x}' \,|\, \mathbf{x}(t) = \delta_{2^n}^i,\, \mathbf{u}(t) = v_i\} = 1. \tag{5.5.6}$$

Otherwise, we can find $v_i \in \mathcal{L}^{2^m}$ in such a way that $a \in \Omega_{k_i-1}(\mathbf{x}')$ when

$$\Pr\{\mathbf{x}(t+1) = a \,|\, \mathbf{x}(t) = \delta_{2^n}^i,\, \mathbf{u}(t) = v_i\} > 0 \;\text{ and }\; a \in \mathcal{L}^{2^n}. \tag{5.5.7}$$

Theorem 5.5.2 (State Feedback)
Consider a P.B.C.N. as in eqn. (5.1.1) and let $\mathbf{x}' = x_1' \ltimes x_2' \ltimes \cdots \ltimes x_n'$. Suppose that the conditions of Thm. 5.5.1 hold. If the structure matrix K of the feedback law has the following shape:

$$K = \begin{bmatrix} v_1 & v_2 & \cdots & v_{2^n} \end{bmatrix}, \tag{5.5.8}$$

where, the vectors v_i are those introduced previously, then the state $\mathbf{x}'$ is globally stable with probability one for the closed-loop system of eqn. (5.5.2).

The following remark is important to understand the usefulness of finding a state feedback controller for stabilisation.

Remark 5.5.1
If a P.B.C.N., as in eqn. (5.1.1), can be globally stabilised to some state $\mathbf{x}'$ with probability one, then every state of the network can be steered to $\mathbf{x}'$ with probability one using random control. Nevertheless, the state feedback scheme of Thm. 5.5.2 is more efficient in terms of stabilisation time than using random control. In fact, the feedback control scheme finds the shortest path to stabilise the network, while a random control scheme finds a path that is always equal or longer than the one found with feedback.

Since the sets $\Omega_k(\mathbf{x}')$ play an important role in the whole stabilisation process, from deciding if the global stabilisation to a state $\mathbf{x}'$ is feasible to determining the actual feedback controller, a simpler way to compute $\Omega_k(\mathbf{x}')$ is needed. This is provided in the next theorem.

Theorem 5.5.3 (Recursive Sets)
Consider a P.B.C.N. as in eqn. (5.1.1). Let L_γ, with $\gamma = 1, \ldots, N$, be the structure matrices of the different B.C.N.s and let L_{in} be as in eqn. (5.5.3). Then, for every $\mathbf{x}' \in \mathcal{L}^{2^n}$, the following holds:

- $\Omega_1(\mathbf{x}') = \{a \in L^{2^n} : \text{there is a}\, \mathbf{u} \in \mathcal{L}^{2^m}\, \text{such that}\, L_{in} \ltimes a \ltimes \mathbf{u} = N\mathbf{x}'\}$;
- $\Omega_{k+1}(\mathbf{x}') = \{a \in L^{2^n} : \text{there is a}\, \mathbf{u} \in \mathcal{L}^{2^m}\, \text{such that}\, L_{in} \ltimes a \ltimes \mathbf{u} = a_1 + \cdots a_N\, \text{for some}\, a_i \in \Omega_k(\mathbf{x}')\, \text{for}\, k = 1, 2, 3, \ldots\}$.

The following remark comments on the results of this theorem.

Remark 5.5.2
From Thm. 5.5.3, one can observe that the structure of $\Omega_k(\mathbf{x}')$ does not depend on the probabilities $P_1, P_2, \ldots, P_N$ that a certain B.C.N. is selected. So, specific values of the selection probabilities do not influence whether a P.B.C.N. can be globally stabilised by state feedback.

Let us now see an example that clarifies the whole procedure of state feedback stabilisation.

Example 5.5.1 (State Feedback)
Consider Ex. 5.1.3. Since, for example, the state δ_4^4 is a control-fixed point, the objective is to find a feedback law, that is a feedback matrix K such that $\mathbf{x}' = \delta_4^4$ is globally stable with probability one for the closed-loop system.

The structure matrices L_γ, with $\gamma = 1, 2, 3, 4$, representing the different B.C.N.s, are similar to those of eqn. (5.1.14). Therefore the matrix L_{in}, defined in eqn. (5.5.3), is built as follows:

$$L_{in} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{5.5.9}$$

Then, the recursive sets of Def. 5.5.1 are found using the procedure explained in Thm. 5.5.3 as follows:

$$\begin{aligned} &\Omega_1(\mathbf{x}') = \{\delta_4^2,\, \delta_4^4\}, \\ &\Omega_2(\mathbf{x}') = \{\delta_4^1,\, \delta_4^2,\, \delta_4^4\}, \\ &\Omega_3(\mathbf{x}') = \mathcal{L}^4. \end{aligned} \tag{5.5.10}$$

Therefore, the two conditions of Thm. 5.5.2 are satisfied.
For $i = 1, 2, 3, 4$, the indices k_i and the inputs v_i are as follows:

$$k_2 = k_4 = 1, \quad k_1 = 2, \quad k_1 = 3, \tag{5.5.11}$$

$$v_1 = \delta_2^1, \quad v_2 = v_3 = v_4 = \delta_2^2. \tag{5.5.12}$$

Then, the state feedback matrix K is:

$$K = \delta_2 \begin{bmatrix} 1 & 2 & 2 & 2 \end{bmatrix}, \tag{5.5.13}$$

which corresponds to the function $\mu = \chi_1 \wedge \chi_2$ and it guarantees that $\mathbf{x}'$ is globally stable with probability one.

The references examined to write this chapter are the following: [6], [13], [14], [15], [26], [27], [31].

Part II

Applications

Chapter 6

Gene Regulatory Networks

The genome of every organism encodes thousands of genes whose products enable cell survival and numerous cellular functions. Genes are fragments of DNA that contain the information about proteins, which are fundamental for the cellular functions. A cell, in fact, is regulated by the interactions between genes and proteins. In order to study these coordinated interactions, studying the collection of genes has proved to be more efficient than studying each individual gene. This is necessary to qualitatively understand the behaviour of complex biological systems. Then, an important role is played by the construction of the model for the simulation of the genetic interactions. The recent literature developed various computational models for these analyses: continuous models, such as linear models and differential equations, which are very precise both in time and in describing molecular concentrations; logical models, which describe the interactions among genes in a qualitative manner and are flexible and easy to adjust to biological phenomena. A logical model that has been recently used is the Boolean Network model, explained in Chapter 2 and first introduced by Kauffman in [1]. Boolean networks are useful to model, simulate and control Gene Regulatory Networks (G.R.N.s). A G.R.N. is a biochemical network that involves genes and proteins. Its role is fundamental for the behaviour of biological organisms since it controls both the internal functions of individual biological cells and the overall development of multicellular organisms. If a Boolean Network is affected by some external boolean inputs, related to an external condition or to the presence of a certain substance in the biological system, it becomes a Boolean Control Network and a control problem can be defined. This is very useful in those situations in which it is desirable to move away from undesired states representing diseases or malfunctions in the system. The applications to this type of network are focused on medical interventions, for example in the treatments of cancer by artificially changing a cell state from cancerous to non-cancerous. In recent years, many applications to gene regulatory networks of biological processes have been simulated and analysed: for example, the cell-fate determination in flower development in [32], the yeast cell-cycle in [33] and the neurotransmitter signaling pathway in [34].
The following chapters will present a few examples of gene regulatory networks modelled with deterministic and probabilistic boolean networks.

The references that inspired the following chapters are the following ones: [13], [15], [35], [36].

Chapter 7

Oxidative Stress Response

In biology, the cause-effect relationships between different pairs of genes are found experimentally. A so-called pathway information exists when these relationships are concatenated together. Biological pathways are used by biologists to represent complex interactions of genes and proteins inside living cells. It can be said that biological pathways represent the graphical interactions between different molecules but they give a marginal picture of the regulation mechanisms among genes and proteins. In order to analyse these complex systems more deeply, genetic regulatory network models consistent with pathway information need to be developed.
Let us start by defining the stress response pathways.

7.1 Stress Response Pathways

Adaptive stress response pathways are the first responders to chemical toxicity, radiation and physical insults. The main architecture that is shared among the different stress response pathways has three main components:

- the transcription factor (TF) is a DNA-binding protein that interacts with the promoter regions of its target genes, via its canonical DNA-binding sites, known as response elements, to activate the expression of the target genes;

- the sensor is a protein that physically interacts with the transcription factor in the cytosol, sequestering the transcription factor from the nucleus under normal cellular conditions;

- the transducer is an enzymatic protein, such as a kinase, that conveys a biochemical change from a signaling pathway upstream of the sensor/TF complex in the event of cellular stress.

It is important to remark that the result of the sensor/TF complexation is to maintain inactivity of the TF under normal cellular conditions, while providing a mechanism that permits activation in response to appropriate insult to the cell. Moreover, generally the sensor and TF are unique for a given stress response pathway unlike transducers which can be common to different stress response pathways. The picture in Fig. 7.1.1 shows the general architecture of a stress response pathway. Let us now focus on the oxidative stress response pathways.

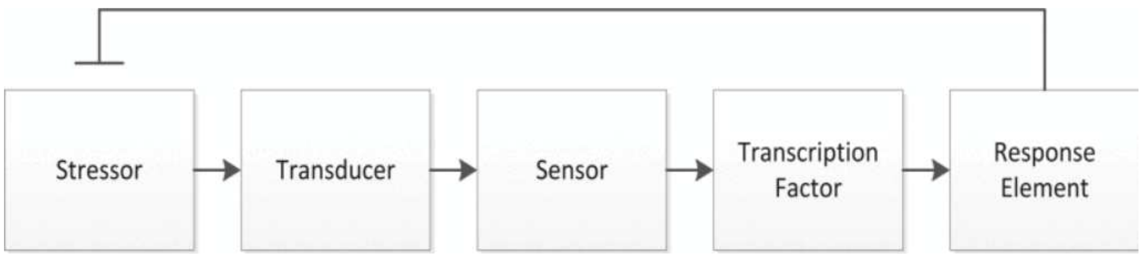

Figure 7.1.1: General scheme of stress response pathways

7.2 Oxidative Stress Response Pathways

Oxidative stress is caused by exposure to reactive oxygen species (ROS). A variety of chronic diseases is due to the stress induced on the cells by electrophiles and oxidants. The result of the interactions between the cell and oxidants is determined largely by the balance between the enzymes that activate the reactive species and the enzymes that detoxify these reactive species. Examples of aging and age-related diseases which are possibly caused by oxidative stress are cancer, cardiovascular disease, chronic inflammation and neurodegenerative disorders. In the following, a counteractive measure to fight oxidative stress is presented.
When the concentration of electrophiles is high, the complex $Keap1 - Nrf2$, composed of the transcription factor $Nrf2$ and sensor $Keap1$ is broken and $Nrf2$ is transported into the nucleus of the cell. Inside, $Nrf2$ forms heterodimers with small Maf proteins (SMP) which then binds to the anti-oxidant response element (ARE). The main purpose of ARE is to detoxify the electrophiles to water soluble components. Therefore, various antioxidant proteins are activated in response to elevated concentrations of electrophiles. The next phase consists in the deactivation of the TF $Nrf2$. $Bach1$ is known as a negative regulator of antioxidant genes through ARE, together with SMP. These protein complexes bind to the same location on the ARE as the $Nrf2-SMP$ complex. Once the electrophiles have been eliminated, $Nrf2$ is transported back to the cytoplasm. At this point, it binds with $Keap1$ which starts its proteosomal degradation.
During a normal metabolism a large number of free radicals are produced. The consequence when the production of the latter goes beyond the maximum level that can be handled by the cellular antioxidant system is oxidative stress, which has an effect in the development of many age-related diseases.

7.3 Boolean Network Modeling

The oxidative stress response pathways can be visually represented via a graph, whose nodes are the main genes involved in the model and whose edges represent either an activation pathway segment if the line ends in an arrow, or an inhibition pathway segment if the line ends in a bar. The genes interactions of the model in [37] are reported in Fig. 7.3.1.

The dynamics involving the main genes can be retrieved from the graph in Fig. 7.3.1, assuming that SMP is ubiquitously expressed, so it is always equal to one.

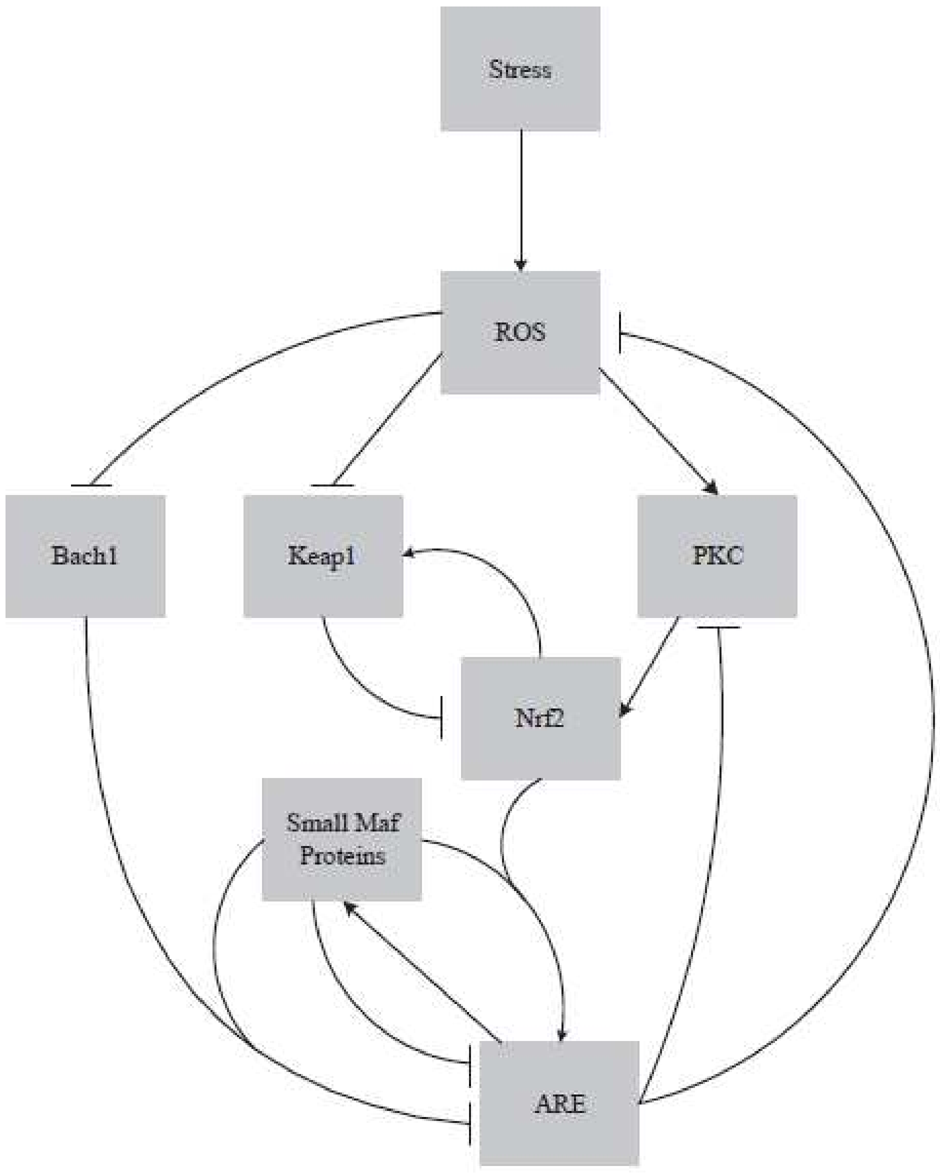

Figure 7.3.1: Oxidative Stress Response Pathways [37]

Then, the update functions for the six genes are collected in the following system:

$$\begin{cases} ROS(t+1) = Stress(t) \wedge \neg ARE(t) \\ Keap1(t+1) = \neg ROS(t) \wedge (Nrf2(t) \vee Keap1(t)) \\ PKC(t+1) = ROS(t) \wedge \neg ARE(t) \\ Nrf2(t+1) = PKC(t) \vee \neg Keap1(t) \\ Bach1(t+1) = \neg ROS(t) \\ ARE(t+1) = Nrf2(t) \wedge (\neg ARE(t) \vee \neg Bach1(t)). \end{cases} \tag{7.3.1}$$

To better understand the system, let us simplify the notation of eqn. (7.3.1) by substituting the genes' names with abstract boolean variables. The substitutions, reported in Tab. 7.3.1, are useful to rewrite eqn. (7.3.1) in a clean form as follows:

Gene	Boolean Variable $\in \mathcal{B}$	Logic Vector $\in \mathcal{L}^2$
$Stress$	$\mu(\cdot)$	$u(\cdot)$
ROS	$\chi_1(\cdot)$	$x_1(\cdot)$
$Keap1$	$\chi_2(\cdot)$	$x_2(\cdot)$
PKC	$\chi_3(\cdot)$	$x_3(\cdot)$
$Nrf2$	$\chi_4(\cdot)$	$x_4(\cdot)$
$Bach1$	$\chi_5(\cdot)$	$x_5(\cdot)$
ARE	$\chi_6(\cdot)$	$x_6(\cdot)$

Table 7.3.1: Table of Conversion of the Oxidative Stress Response Model

$$\begin{cases} \chi_1(t+1) = \mu(t) \wedge \neg\chi_6(t) \\ \chi_2(t+1) = \neg\chi_1(t) \wedge (\chi_4(t) \vee \chi_2(t)) \\ \chi_3(t+1) = \chi_1(t) \wedge \neg\chi_6(t) \\ \chi_4(t+1) = \chi_3(t) \vee \neg\chi_2(t) \\ \chi_5(t+1) = \neg\chi_1(t) \\ \chi_6(t+1) = \chi_4(t) \wedge (\neg\chi_6(t) \vee \neg\chi_5(t)). \end{cases} \tag{7.3.2}$$

In order to find the algebraic form and the transition matrix L of this system of equations, it is necessary to translate the boolean variables into logic vectors and the boolean operators into logic matrices as follows:

$$\begin{cases} x_1(t+1) = M_\wedge \ltimes u(t) \ltimes M_\neg \ltimes x_6(t) \\ x_2(t+1) = M_\wedge \ltimes M_\neg \ltimes x_1(t) \ltimes M_\vee \ltimes x_4(t) \ltimes x_2(t) \\ x_3(t+1) = M_\wedge \ltimes x_1(t) \ltimes M_\neg \ltimes x_6(t) \\ x_4(t+1) = M_\vee \ltimes x_3(t) \ltimes M_\neg \ltimes x_2(t) \\ x_5(t+1) = M_\neg \ltimes x_1(t) \\ x_6(t+1) = M_\wedge \ltimes x_4(t) \ltimes M_\vee \ltimes M_\neg \ltimes x_6(t) \ltimes M_\neg \ltimes x_5(t). \end{cases} \tag{7.3.3}$$

Now, by applying the theory of B.C.N.s, it is possible to derive the transition matrix L from eqn. (7.3.3):

$$L = \delta_{64} \begin{bmatrix} 64 & 23 & 63 & 23 & 64 & \cdots & 62 & 62 & 62 & 62 \end{bmatrix} \in \mathcal{L}^{128\times 64}. \tag{7.3.4}$$

Since a boolean input corresponding to the $Stress$ gene is present in the system, one can divide the transition matrix L into two structure matrices, L_1 and L_2:

$$\begin{cases} L_1 = \delta_{64}\ [64\ 23\ 63\ 23\ 64\ 24\ 64\ 24\ 64\ 23\ 63\ 23\ 64\ 24\ 64\ 24\ 60\ 19\ 59\ 19\ 60\ 20\ 60\ 20\ 64\ 23\ 63\ 23\ 64\ 24\ 64\ 24\ \ldots \\ \qquad 46\ 13\ 45\ 13\ 46\ 14\ 46\ 14\ 46\ 13\ 45\ 13\ 46\ 14\ 46\ 14\ 42\ 9\ 41\ 9\ 58\ 26\ 58\ 26\ 46\ 13\ 45\ 13\ 62\ 30\ 62\ 30] \in \mathcal{L}^{64\times 64} \\ L_2 = \delta_{64}\ [64\ 55\ 63\ 55\ 64\ 56\ 64\ 56\ 64\ 55\ 63\ 55\ 64\ 56\ 64\ 56\ 60\ 51\ 59\ 51\ 60\ 52\ 60\ 52\ 64\ 55\ 63\ 55\ 64\ 56\ 64\ 56\ \ldots \\ \qquad 46\ 45\ 45\ 45\ 46\ 46\ 46\ 46\ 46\ 45\ 45\ 45\ 46\ 46\ 46\ 46\ 42\ 41\ 41\ 41\ 58\ 58\ 58\ 58\ 46\ 45\ 45\ 45\ 62\ 62\ 62\ 62] \in \mathcal{L}^{64\times 64}. \end{cases} \tag{7.3.5}$$

The input $u(t)$, whose value discriminates the choice between the structure matrices L_1 and L_2, cannot be arbitrarily controlled since it depends on the level of oxidative stress induced on the cells. Therefore, instead of studying the whole B.C.N., it is still possible to analyse the oxidative stress response in the two separate B.N.s

corresponding to no stress (L_2) and stress (L_1).
Let us start with the analysis of L_2, when the input $u(t) = Stress(t) = 0$, $\forall t \geq 0$. The state graph of this B.N. is reported in Fig. 7.3.4 and the fixed point δ_{64}^{46} is highlighted by the coloured box. Since there is a unique stable point, it is possible to observe in the graph that all the other nodes converge towards the state δ_{64}^{46} in a finite number of steps, so the B.N. is globally stable. Equivalently, all the other nodes belong to its basin of attraction. The states vector of the system in eqn. (7.3.1) is $\begin{bmatrix} ROS & Keap1 & PKC & Nrf2 & Bach1 & ARE \end{bmatrix}$. Then, the logic vector δ_{64}^{46} corresponds, using Prop. 1.3.3, to the following state vector: $\begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$. This means that both $Keap1$ and $Bach1$ are active, while the other genes/proteins are off. These results are compatible with the oxidative stress response mechanism since $Keap1$ is the sensor that detects the presence of electrophiles and oxidants, while $Bach1$ is the inhibitor of antioxidant genes through ARE. In other words, $Keap1$ is responsible for the start of the oxidative stress response and $Bach1$ blocks this mechanism when there are no oxidative elements.
Then, the boolean variables, in this specific case, are:

$$\begin{cases} \chi_1(t) = 0 \\ \chi_2(t) = 1 \\ \chi_3(t) = 0 \\ \chi_4(t) = 0 \\ \chi_5(t) = 1 \\ \chi_6(t) = 0. \end{cases} \qquad \forall t \tag{7.3.6}$$

Following the theory of Boolean Networks, explained in Chapter 2, a deeper analysis of this particular model can be made.
Checking the fixed points and limit cycles with the formulas of Thm. 2.3.1, Thm. 2.3.2 and Prop. 2.3.2, one finds a single fixed point represented by the state δ_{64}^{46}. The latter is also the only attractor of the B.N.. Then, the absorption time of the network, defined in Def. 2.4.1, is the absorption time of δ_{64}^{46}. By the equivalence between the absorption time and the exponent r_0 in Thm. 2.4.1, it is possible to prove that the maximum number of steps to reach the fixed point δ_{64}^{46} is equal to $r_0 = 4$.
Since there is only one attractor, its basin of attraction contains all the other states of the network, with no need to compute it.
In terms of communication classes, as defined in Defs. 2.5.5 and 2.5.6, the fixed point is a closed class while every other state belongs to a transient class.
Finally, the stability of the B.N. can be analysed. As previously mentioned, this network is globally stable since all states converge to δ_{64}^{46} in a finite number of steps. This can be observed in the state graph in Fig. 7.3.4. However, to correctly prove the global stability, one of the two conditions of Thm. 2.6.1 must be verified. With a few lines of code it is possible to obtain the powers of the transition matrix and check that the second condition holds, thus proving the global stability of the B.N..
In the following, a few simulations of the B.N. model are presented, showing the time response behaviour in terms of both the boolean variables and the logic vectors. The initial condition of the system is $\begin{bmatrix} ROS & Keap1 & PKC & Nrf2 & Bach1 & ARE \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$. Fig. 7.3.2 shows the variables evolution due to input $u(t) = Stress(t) = 0$, $\forall t = 1, 2 \ldots, 100$. It is possible to observe that, after a transitory of

four time steps, the boolean variables converge to the stable configuration at $t = 5$. The same behaviour can be noticed looking at Fig. 7.3.3, which shows the state evolution and the convergence to the fixed point δ_{64}^{46}.

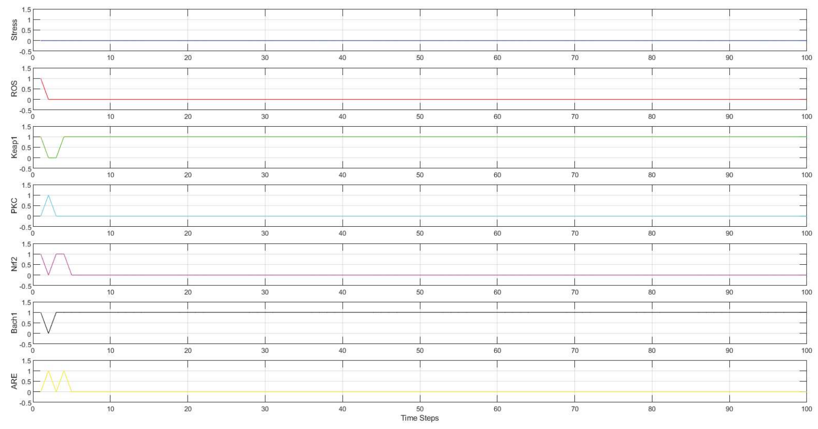

Figure 7.3.2: Time response behaviour of the system with $u(t) = Stress(t) = 0$

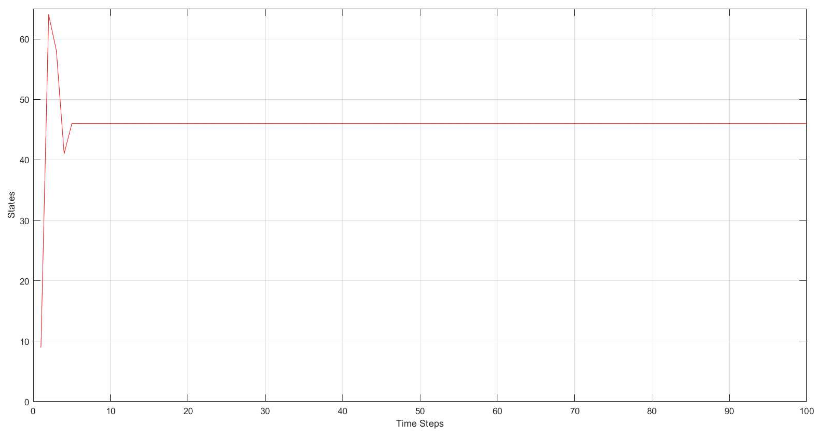

Figure 7.3.3: Time response behaviour of the system with $u(t) = Stress(t) = 0$

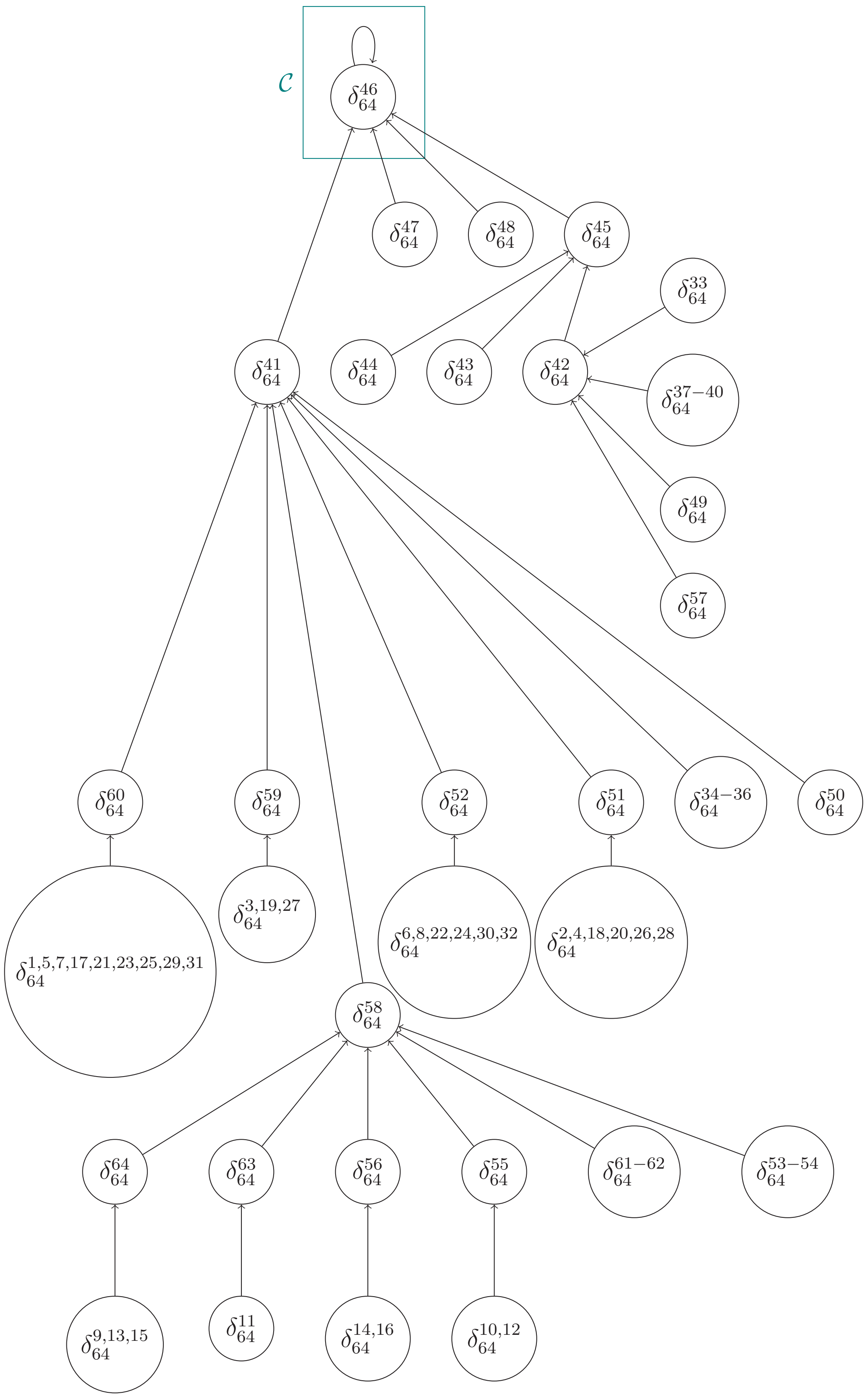

Figure 7.3.4: Oxidative Stress Response with $Stress = 0$

Let us now analyse L_1, that represents the system matrix in the presence of oxidative stress or equivalently when the input $u(t) = Stress(t) = 1$, $\forall t$. The state graph of this B.N. is reported in Fig. 7.3.7. It is possible to observe an oscillatory behaviour of the genes represented by a limit cycle, highlighted by the coloured box, which is composed of the following seven states: δ_{64}^{46}, δ_{64}^{14}, δ_{64}^{24}, δ_{64}^{20}, δ_{64}^{19}, δ_{64}^{59}, δ_{64}^{41}.
Then, the boolean variables behaviour, in this specific case, is the following:

$$\begin{cases} \chi_1(1)=0\\ \chi_2(1)=1\\ \chi_3(1)=0\\ \chi_4(1)=0\\ \chi_5(1)=1\\ \chi_6(1)=0 \end{cases} \rightarrow \begin{cases} \chi_1(2)=1\\ \chi_2(2)=1\\ \chi_3(2)=0\\ \chi_4(2)=0\\ \chi_5(2)=1\\ \chi_6(2)=0 \end{cases} \rightarrow \begin{cases} \chi_1(3)=1\\ \chi_2(3)=0\\ \chi_3(3)=1\\ \chi_4(3)=0\\ \chi_5(3)=0\\ \chi_6(3)=0 \end{cases} \rightarrow \begin{cases} \chi_1(4)=1\\ \chi_2(4)=0\\ \chi_3(4)=1\\ \chi_4(4)=1\\ \chi_5(4)=0\\ \chi_6(4)=0 \end{cases} \rightarrow$$
$$\rightarrow \begin{cases} \chi_1(5)=0\\ \chi_2(5)=0\\ \chi_3(5)=1\\ \chi_4(5)=1\\ \chi_5(5)=0\\ \chi_6(5)=1. \end{cases} \rightarrow \begin{cases} \chi_1(6)=0\\ \chi_2(6)=0\\ \chi_3(6)=0\\ \chi_4(6)=1\\ \chi_5(6)=0\\ \chi_6(6)=1 \end{cases} \rightarrow \begin{cases} \chi_1(7)=0\\ \chi_2(7)=1\\ \chi_3(7)=0\\ \chi_4(7)=1\\ \chi_5(7)=1\\ \chi_6(7)=1 \end{cases} \rightarrow \begin{cases} \chi_1(8)=0\\ \chi_2(8)=1\\ \chi_3(8)=0\\ \chi_4(8)=0\\ \chi_5(8)=1\\ \chi_6(8)=0 \end{cases} \tag{7.3.7}$$

Let us now explain why this limit cycle is consistent with the scheme of the oxidative stress response pathways, analysing each state of the cycle:

1. δ_{64}^{46} represents the aforementioned condition of a disabled cycle (L_2);
2. with the input $u(t) = 1$, $\forall t$, the state δ_{64}^{14} differs from the previous one because electrophiles and oxidants ($\chi_1(2) = ROS(2)$) are present in the cell;
3. at δ_{64}^{24}, due to the elevated concentrations of electrophiles, the complex $Keap1-Nrf2$ is broken and the transducer PKC (protein kinase C) is activated;
4. the state δ_{64}^{20} indicates that the transcription factor $Nrf2$ enters the cell nucleus;
5. at δ_{64}^{19}, the antioxidant response element (ARE), made up of various antioxidant proteins, detoxifies the electrophiles;
6. at δ_{64}^{59}, once the electrophiles have been neutralized, the transducer PKC is deactivated to stop the translation of the ARE;
7. δ_{64}^{41} is useful to reactivate the sensor $Keap1$ and the inhibitor $Bach1$, in case new oxidant elements enter the cell;
8. δ_{64}^{46} corresponds to the first state, thus the limit cycle is over.

Let us now analyse the system more deeply, as we did for the B.N. L_2.
Checking the fixed points and limit cycles with the formulas of Thm. 2.3.1, Thm. 2.3.2 and Prop. 2.3.2, one finds a single limit cycle consisting of the following set of states: $\{\delta_{64}^{46}, \delta_{64}^{14}, \delta_{64}^{24}, \delta_{64}^{20}, \delta_{64}^{19}, \delta_{64}^{59}, \delta_{64}^{41}\}$. This is the only attractor in the B.N.. Therefore, the absorption time of the network is the maximum number of steps to reach one of the seven states forming the limit cycle. By using Thm. 2.4.1, the

absorption time is $r_0 = 7$.
As in the previous network, the basin of attraction of the limit cycle contains all the states of this B.N. because it is the only attractor.
The states of the limit cycle belong to a closed class, while every other state is a transient state.
Since there is no fixed point, the B.N. is not globally stable. In fact, even though the limit cycle is reached from every possible initial condition, the state changes at each time instant. In this case, the limit cycle is called a globally attractive cycle, as in Prop. 2.6.1.
In the following, a few simulations of the B.N. model are presented, showing the time response behaviour in terms of both the boolean variables and the logic vectors. The initial condition of the system is again $\begin{bmatrix} ROS & Keap1 & PKC & Nrf2 & Bach1 & ARE \end{bmatrix}$ $= \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$. Fig. 7.3.5 shows the variables evolution due to input $u(t) = Stress(t) = 1$, $\forall t = 1, 2 \ldots, 100$. It is possible to observe that, after a small transitory, the boolean variables enter an oscillatory behaviour. The same behaviour can be noticed looking at Fig. 7.3.6, which shows the state evolution and the oscillation of the states $\{\delta_{64}^{46}, \delta_{64}^{14}, \delta_{64}^{24}, \delta_{64}^{20}, \delta_{64}^{19}, \delta_{64}^{59}, \delta_{64}^{41}\}$ according to the limit cycle.

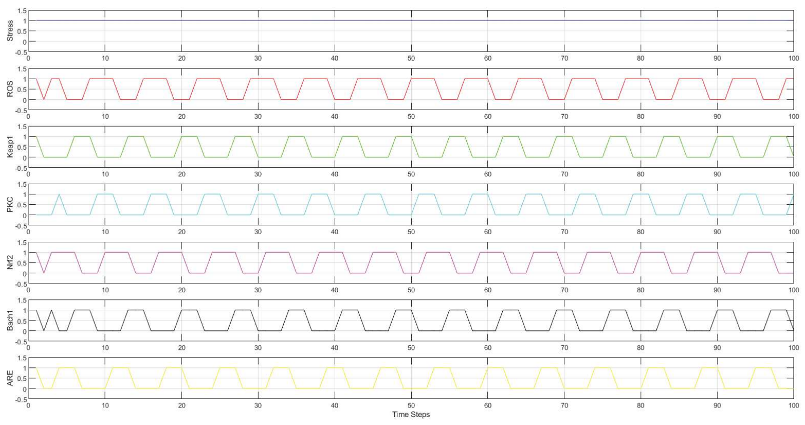

Figure 7.3.5: Time response behaviour of the system with $u(t) = Stress(t) = 1$

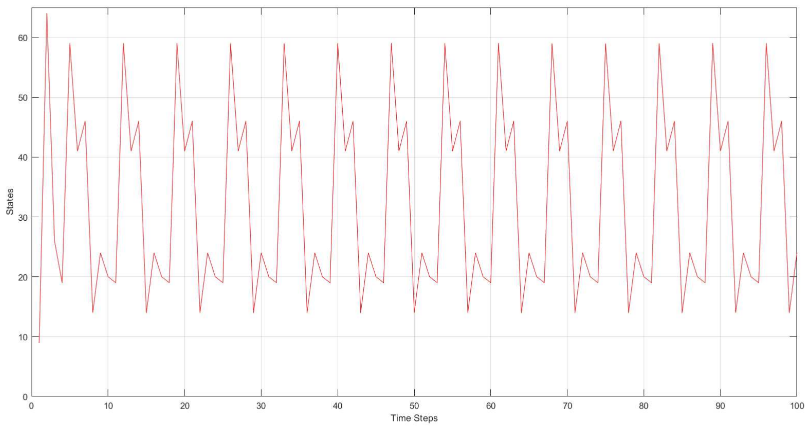

Figure 7.3.6: Time response behaviour of the system with $u(t) = Stress(t) = 1$

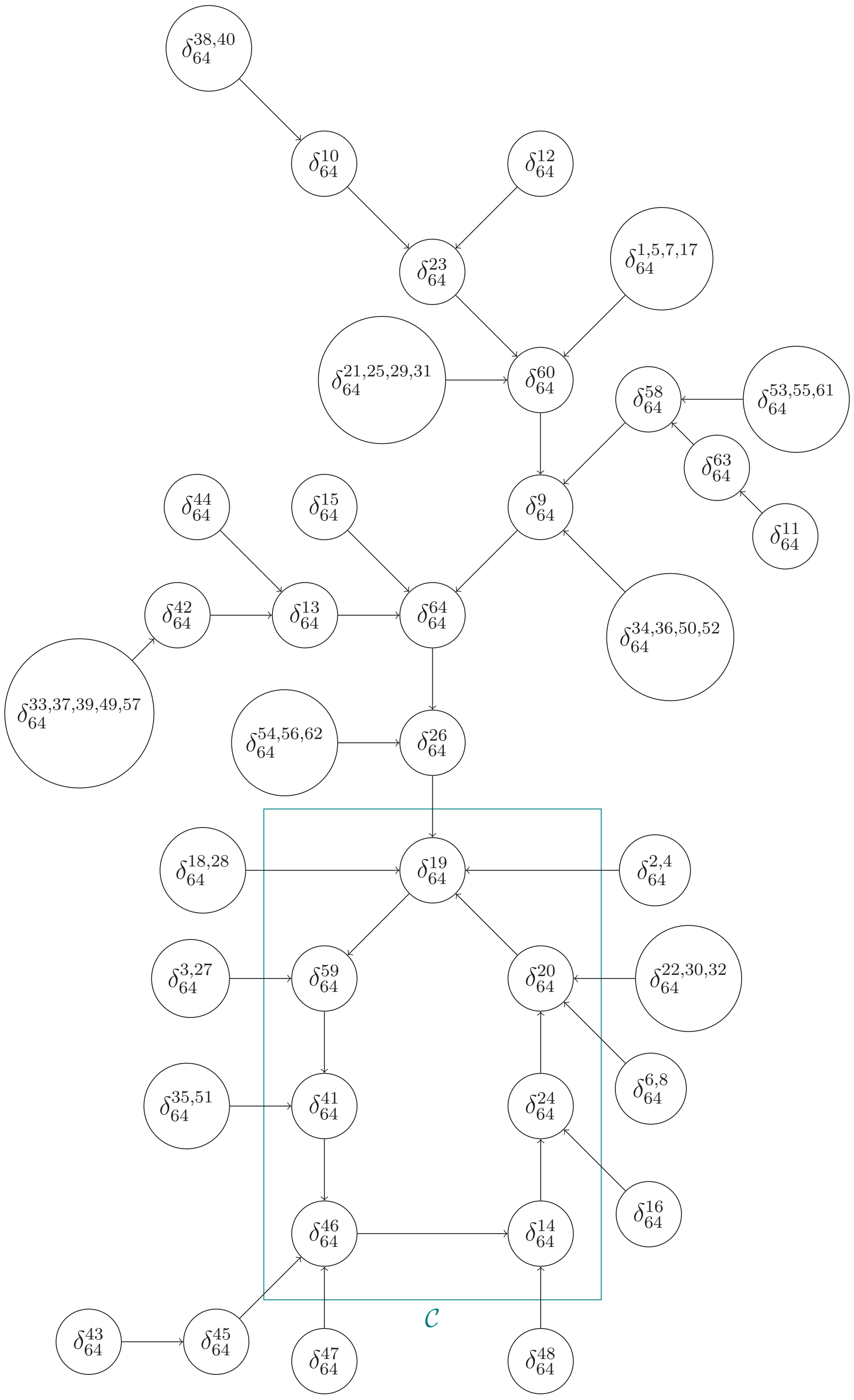

Figure 7.3.7: Oxidative Stress Response with $Stress = 1$

Finally, one can observe the time response behaviour of the entire B.C.N. by mixing the two values of the input. The evolution of both the boolean variables and the state vectors is reported in Figs. 7.3.8 and 7.3.9, with initial condition $\begin{bmatrix} ROS & Keap1 & PKC & Nrf2 & Bach1 & ARE \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$ and input $u(t) = 0$, for $1 \leq t \leq 24$ and $75 \leq t \leq 100$, and $u(t) = 1$, for $25 \leq t \leq 74$. The results are compatible with those of [37], confirming that the model is correct.

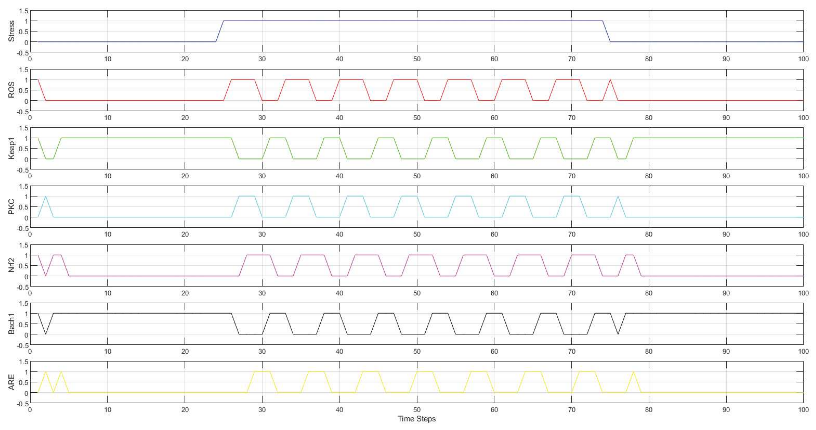

Figure 7.3.8: Time response behaviour of the system with variable $u(t) = Stress(t)$

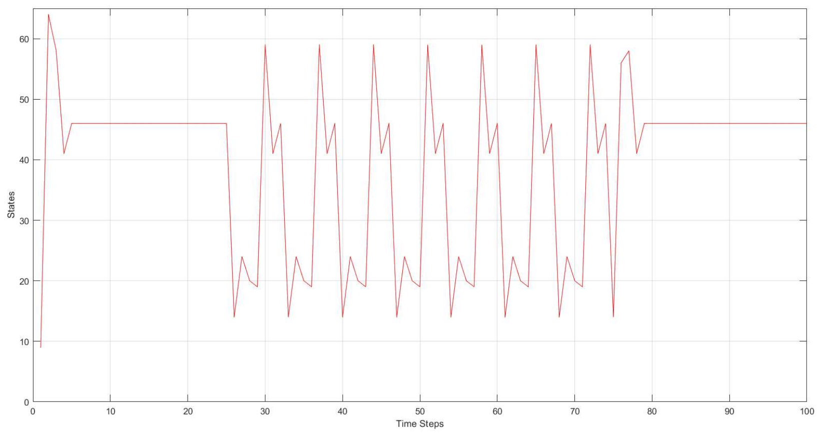

Figure 7.3.9: Time response behaviour of the system with variable $u(t) = Stress(t)$

The references examined to write this chapter are the following: [13], [15], [35], [36], [37], [38], [39], [40].

Chapter 8

Early Detection of Cancer

Many diseases are the results of interactions of genes according to their regulatory rules. For example, cancers are caused by changes in the gene expression program inside individual cells.
In this chapter, the construction of a P.B.N. for the hepatocellular cancer GRN is presented. Starting from the 27 cancer genes discovered in [41], the study in [42] uses a regression analysis to identify the regulatory networks in the two cases of tumor and non-tumor networks. The result of this analysis is the selection of a subgroup of 12 genes for each network, both containing the three cancer genes GS1706, GS1938 and GS2508.
Now, the goal is to build the P.B.N.s of the two systems and test that the results are compatible with those obtained in the original study (see [42]).

8.1 Probabilistic Boolean Network Modeling

8.1.1 Probabilistic Boolean Network of tumor cells

Let us start with the P.B.N. in the case of tumor cells.
The system of equations describing the P.B.N. is the following:

$$\begin{cases}
G_1(t+1) = f_1^1\left(G_5(t),\, G_6(t),\, G_{12}(t)\right) = \\
\qquad = (\neg G_5(t) \wedge (\neg G_6(t) \wedge G_{12}(t)))\vee \\
\qquad \vee(G_5(t) \wedge (\neg G_6(t) \vee G_{12}(t))), \quad p_1^1 = 1 \\
G_2(t+1) = f_2^1\left(G_7(t),\, G_{10}(t),\, G_{11}(t)\right) = G_{11}(t), \quad p_2^1 = 1 \\
G_3(t+1) = f_3^1\left(G_4(t),\, G_8(t),\, G_9(t)\right) = \\
\qquad = (\neg G_8(t) \wedge (G_4(t) \wedge G_9(t)))\vee \\
\qquad \vee(G_8(t) \wedge (G_4(t) \vee G_9(t))), \quad p_3^1 = 1 \\
G_4(t+1) = f_4\left(G_8(t),\, G_{10}(t)\right) = \\
\qquad = \begin{cases} f_4^1\left(G_8(t)\right) = G_8(t), \quad p_4^1 = 0.58 \\ f_4^2\left(G_{10}(t)\right) = G_{10}(t), \quad p_4^2 = 0.42 \end{cases} \\
G_5(t+1) = f_5^1\left(G_1(t),\, G_6(t),\, G_{12}(t)\right) = \\
\qquad = \neg G_6(t) \wedge (\neg G_{12}(t) \wedge G_1(t)) \vee G_6(t), \quad p_5^1 = 1 \\
G_6(t+1) = f_6^1\left(G_1(t),\, G_5(t),\, G_{12}(t)\right) = \\
\qquad = (\neg G_{12(t)} \wedge (\neg G_1(t) \wedge G_5(t))) \vee (G_{12}(t) \wedge G_5(t)), \quad p_6^1 = 1 \\
G_7(t+1) = f_7^1\left(G_{10}(t)\right) = G_{10}(t), \quad p_7^1 = 1 \\
G_8(t+1) = f_8^1\left(G_4(t)\right) = G_4(t), \quad p_8^1 = 1 \\
G_9(t+1) = f_9^1\left(G_3(t)\right) = G_3(t), \quad p_9^1 = 1 \\
G_{10}(t+1) = f_{10}\left(G_1(t),\, G_5(t),\, G_6(t),\, G_7(t),\, G_{12}(t)\right) = \\
\qquad = \begin{cases} f_{10}^1\left(G_7(t)\right) = G_7(t), \quad p_{10}^1 = 0.34 \\ f_{10}^2\left(G_1(t),\, G_5(t),\, G_6(t)\right) = \\ \quad = (\neg G_1(t) \wedge (G_5(t) \vee G_6(t)))\vee \\ \quad \vee(G_1(t) \wedge (G_5(t) \wedge G_6(t))), \quad p_{10}^2 = 0.33 \\ f_{10}^3\left(G_5(t),\, G_6(t),\, G_{12}(t)\right) = \\ \quad = (\neg G_5(t) \wedge (G_6(t) \wedge \neg G_{12}(t)))\vee \\ \quad \vee(G_5(t) \wedge (G_6(t) \vee \neg G_{12}(t))), \quad p_{10}^3 = 0.33 \end{cases} \\
G_{11}(t+1) = f_{11}^1\left(G_{11}(t)\right) = G_{11}(t), \quad p_{11}^1 = 1 \\
G_{12}(t+1) = f_{12}\left(G_1(t),\, G_2(t),\, G_5(t),\, G_6(t)\right) = \\
\qquad = \begin{cases} f_{12}^1\left(G_1(t),\, G_5(t),\, G_6(t)\right) = \\ \quad = (\neg G_1(t) \wedge (\neg G_5(t) \wedge G_6(t)))\vee \\ \quad \vee(G_1(t) \wedge (\neg G_5 \vee G_6(t))), \quad p_{12}^1 = 0.55 \\ f_{12}^2\left(G_2(t)\right) = G_2(t), \quad p_{12}^2 = 0.45. \end{cases}
\end{cases} \tag{8.1.1}$$

First, let us prove that the results obtained in [42] are correct by running some simple simulations in Matlab.

Starting from the initial condition

$$\begin{bmatrix} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 & G_7 & G_8 & G_9 & G_{10} & G_{11} & G_{12} \end{bmatrix} =$$
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

the evolution of the system after 500 iterations is shown in Fig. 8.1.1, while the evolution of the three cancer genes is shown in Fig. 8.1.2. One can observe that the three cancer genes expression continues to oscillate throughout the simulation.

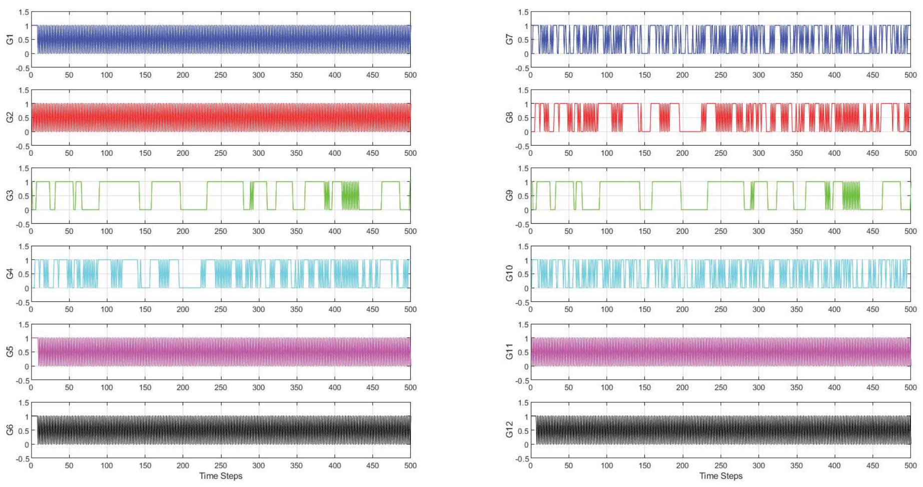

Figure 8.1.1: Time evolution of the system in tumor networks

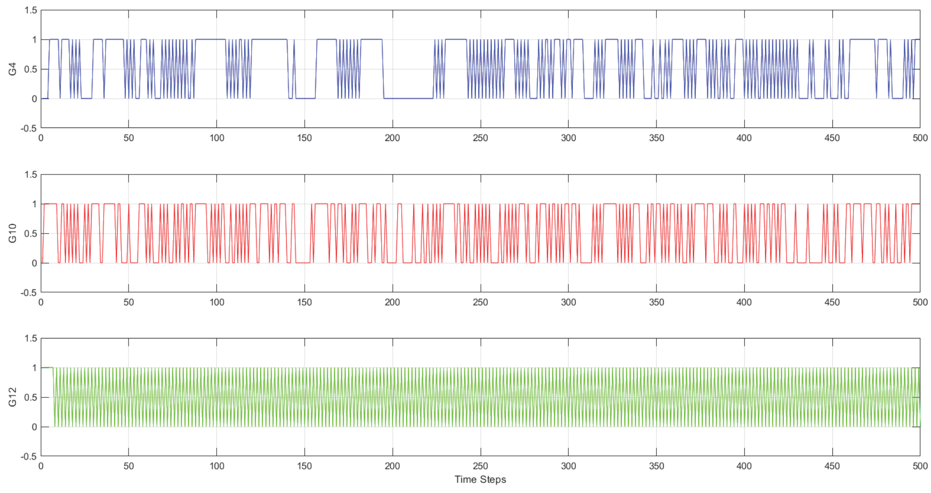

Figure 8.1.2: Time evolution of the three cancer genes in tumor networks

The next graph, presented in Fig. 8.1.3, is obtained by running 1000 simulations with 4000 iterations each, and random initial condition (corresponding to situations where the cancer genes are unexpressed), and by counting the number of times the three cancer states end up in each of the eight possible combinations. The graph shows that the desired combination (0, 0, 0), corresponding to the stable case when all three cancer genes $G4$, $G10$ and $G12$ are unexpressed, occurs only about 11% of the times. In the remaining 89% of the cases at least one cancer gene is expressed and hence the cells evolve into cancerous cells.

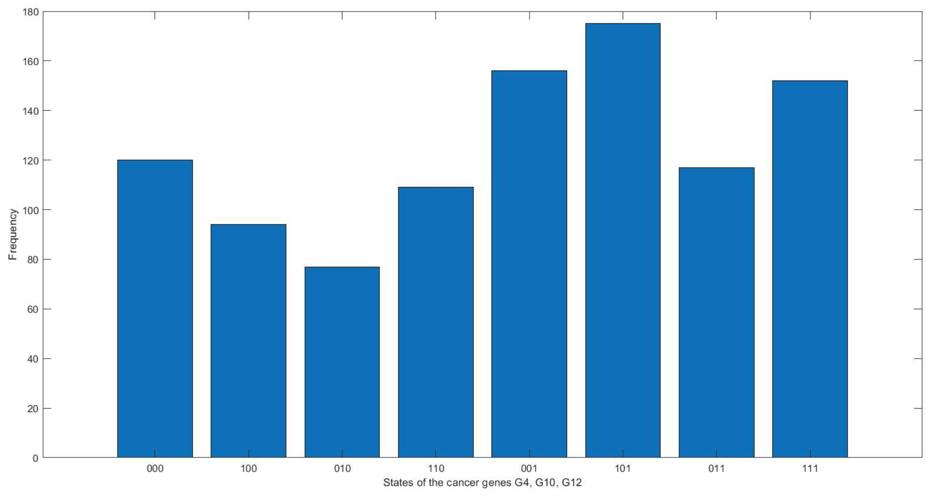

Figure 8.1.3: Steady state behaviour of the three cancer genes in tumor networks

In order to convert the logic system into its algebraic form, one needs to know the number of networks of the P.B.N., that is the number of possible combinations of logic functions and it is equal to $N = \prod_{i=1}^{12} l_i = l_1 \cdot \ldots \cdot l_{12} = 1 \cdot \ldots \cdot l_4 \cdot l_{10} \cdot l_{12} = 1 \cdot \ldots \cdot 2 \cdot 3 \cdot 2 = 12$, as explained in Prop. 4.1.1.
The structure matrices of the twelve B.N.s are computed by using a function provided by Cheng in [7] which automatically transforms the logic equations in algebraic form and evaluates the Khatri-Rao product between the functions structure matrices.
The twelve structure matrices and their corresponding probabilities are listed below:

$$\begin{aligned}
L_1 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^1_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}},\\
&\qquad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^1_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.1085\\
L_2 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^1_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}},\\
&\qquad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^1_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0887\\
L_3 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^2_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}},\\
&\qquad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^2_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.1053\\
L_4 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^2_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}},\\
&\qquad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^2_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0861
\end{aligned}$$

$$
\begin{aligned}
L_5 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^3_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^3_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.1053 \\
L_6 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^1_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^3_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^1_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^3_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0861 \\
L_7 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^1_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^1_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.0785 \\
L_8 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^1_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^1_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0643 \\
L_9 &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^2_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^2_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.0762 \\
L_{10} &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^2_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^2_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0624 \\
L_{11} &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^3_{f_{10}} * M^1_{f_{11}} * M^1_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^3_{10} \cdot p^1_{11} \cdot p^1_{12} = 0.0762 \\
L_{12} &= M^1_{f_1} * M^1_{f_2} * M^1_{f_3} * M^2_{f_4} * M^1_{f_5} * M^1_{f_6} * M^1_{f_7} * M^1_{f_8} * M^1_{f_9} * M^3_{f_{10}} * M^1_{f_{11}} * M^2_{f_{12}}, \\
&\quad \text{with } P_1 = p^1_1 \cdot p^1_2 \cdot p^1_3 \cdot p^2_4 \cdot p^1_5 \cdot p^1_6 \cdot p^1_7 \cdot p^1_8 \cdot p^1_9 \cdot p^3_{10} \cdot p^1_{11} \cdot p^2_{12} = 0.0624.
\end{aligned}
\tag{8.1.2}
$$

It is now possible to simulate the evolution of this P.B.N. using the algebraic form, and hence the twelve structure matrices. In Figs. 8.1.4 and 8.1.5, the time evolution of the states and the time evolution of the three cancer genes are reported. Both graphs confirm the fact that the algebraic form of the P.B.N. reflects the dynamics of the logic system with an oscillatory behaviour.

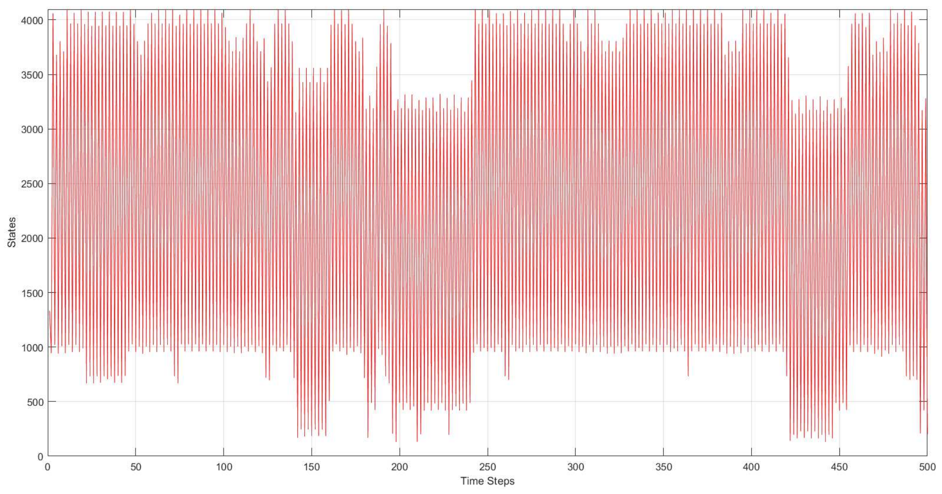

Figure 8.1.4: Time evolution of the states in tumor networks

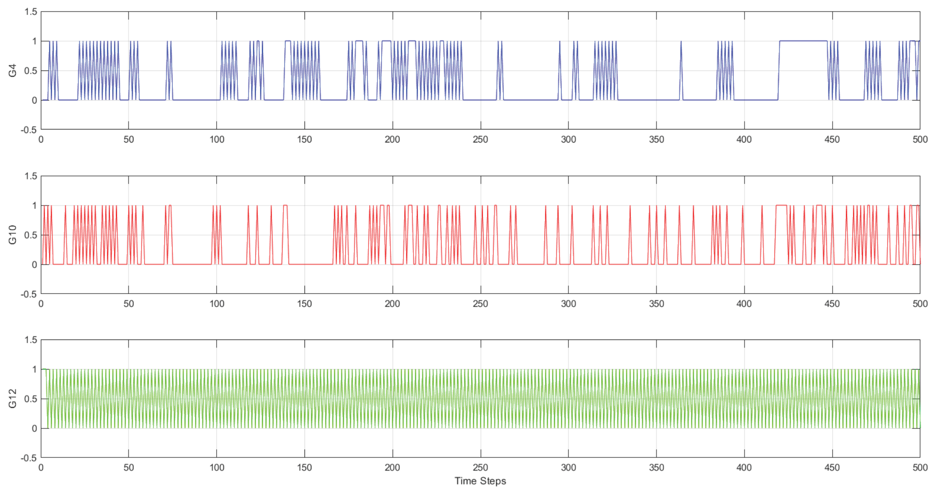

Figure 8.1.5: Time evolution of the three cancer genes in tumor networks

From the update equations and using Def. 4.2.3, it is possible to derive the incidence matrix as in the following:

$$\mathcal{I}(\mathbf{G}) = \bigvee_{\gamma=1}^{12} \mathcal{I}(\mathbf{G},\, \Sigma_\gamma) = \begin{bmatrix} 0&0&0&0&1&1&0&0&0&1&0&1\\ 0&0&0&0&0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&0&0&1&0&0&0\\ 0&0&1&0&0&0&0&1&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&1&0&1\\ 1&0&0&0&1&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&0\\ 0&0&1&1&0&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&1&0&0&1&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&0&0&0&0\\ 1&0&0&0&1&1&0&0&0&1&0&0 \end{bmatrix} \tag{8.1.3}$$

From the latter, it is easy to draw the graph of this P.B.N., which is shown in Fig. 8.1.6. The three cancer genes are highlighted in different colours.
The state graph of this P.B.N., instead, is too complex to be drawn because it is composed of $2^{12} = 4096$ states and $2^{12} \cdot 12 = 49152$ edges.

In the following, an analysis of the fixed points, limit cycles and stability of the P.B.N. is presented.
The number of fixed points for each of the twelve B.N.s is as follows:

$$N_p = \begin{bmatrix} 24 & 12 & 12 & 6 & 12 & 6 & 12 & 6 & 6 & 3 & 6 & 3 \end{bmatrix}.$$

Among these fixed points, there is only one common fixed point (see Def. 4.3.2): $\delta_{p_c} = \delta_{4096}^{4096}$, which means that all genes are unexpressed. Now, in a simulation, it is not possible to reach this state unless in the trivial case where the initial condition is exactly the state δ_{4096}^{4096}. The other fixed points can be reached but, since they are fixed points of only one or a subset of the structure matrices, they are not a closed

class, thus the probability to move to another state is different from zero. This explains the oscillatory behaviour of the P.B.N. when it is expressed in its algebraic form.
For what regards the limit cycles, it is computationally possible to evaluate the limit cycles considering a single structure matrix. Instead, the limit cycles that involve multiple structure matrices require high computational power due to the sizes of these matrices. In fact, the longer the simulation the higher is the number of combinations of these matrices: at the first step, one can choose among 12 matrices, at the second step among 12 matrices and so on, leading to an exponential growth. The matrix for the global stability with probability one of P.B.N.s, presented in Thm. 4.4.1, is computationally easier to obtain. In this case, no matrix row is unitary. Therefore, one can claim that there is no global stability with probability one in this network, as observed in the previous simulations.

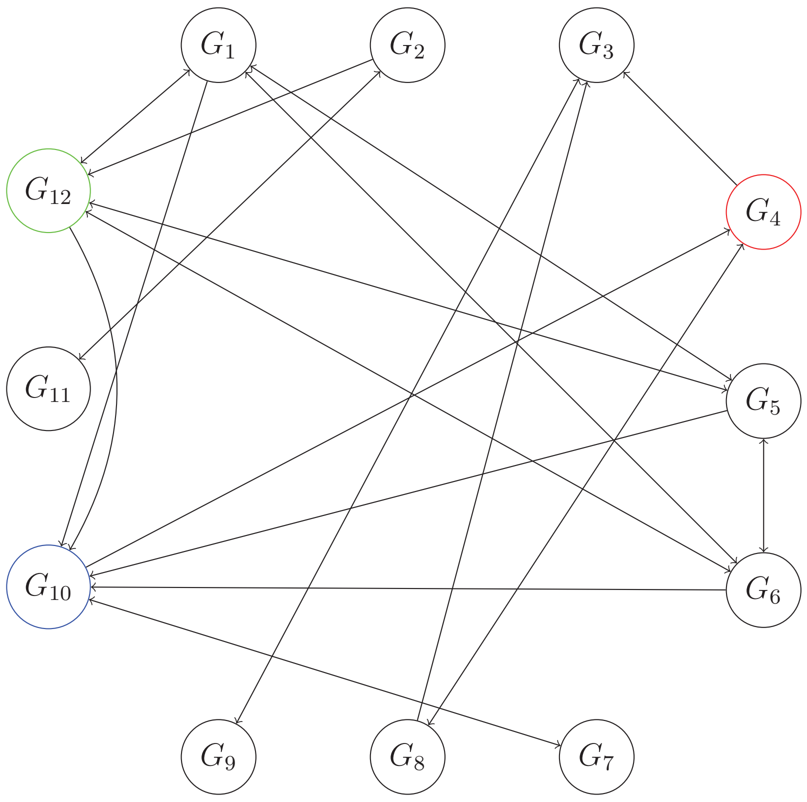

Figure 8.1.6: Graph of the Boolean Network for tumor cells

8.1.2 Probabilistic Boolean Network of non-tumor cells

Let us now focus on the P.B.N. in the case of non-tumor cells.
The system of equations describing the P.B.N. is the following:

$$\begin{cases}
g_1(t+1) = f_1^1\left(g_8(t)\right) = g_8(t), \quad p_1^1 = 1 \\
g_2(t+1) = f_2^1\left(g_{12}(t)\right) = g_{12}(t), \quad p_2^1 = 1 \\
g_3(t+1) = f_3^1\left(g_4(t),\, g_5(t),\, g_7(t)\right) = \\
\qquad = (\neg g_7(t) \wedge (\neg g_4(t) \wedge g_5(t))) \vee (g_7(t) \wedge (\neg g_4(t) \vee g_5(t))), \quad p_3^1 = 1 \\
g_4(t+1) = f_4^1\left(g_5(t),\, g_6(t),\, g_7(t)\right) = \\
\qquad = (\neg g_5(t) \wedge (\neg g_6(t) \wedge g_7(t))) \vee (g_7(t) \wedge g_5(t)), \quad p_4^1 = 1 \\
g_5(t+1) = f_5^1\left(g_4(t),\, g_6(t),\, g_7(t)\right) = \\
\qquad = (\neg g_6(t) \wedge (\neg g_7(t) \wedge g_4(t)) \vee (g_6(t) \wedge (\neg g_7(t) \vee g_4(t))), \quad p_5^1 = 1 \\
g_6(t+1) = f_6^1\left(g_7(t)\right) = g_7(t), \quad p_6^1 = 1 \\
g_7(t+1) = f_7\left(g_1(t),\, g_3(t),\, g_4(t),\, g_5(t),\, g_6(t),\, g_8(t),\, g_{10}(t)\right) = \\
\qquad = \begin{cases}
f_7^1\left(g_4(t),\, g_5(t),\, g_6(t)\right) = \\
\quad = (\neg g_4(t) \wedge g_6(t)) \vee (g_4(t) \wedge (\neg g_5(t) \vee g_6(t))), \quad p_7^1 = 0.37 \\
f_7^2\left(g_3(t),\, g_8(t),\, g_{10}(t)\right) = \\
\quad = (\neg g_3(t) \wedge (\neg g_8(t) \wedge g_{10}(t))) \vee \\
\quad \vee (g_3(t) \wedge (\neg g_8(t) \vee g_{10}(t))), \quad p_7^2 = 0.32 \\
f_7^3\left(g_1(t),\, g_3(t),\, g_8(t)\right) = \\
\quad = (\neg g_1(t) \wedge g_3(t)) \vee (g_1(t) \wedge (\neg g_8(t) \vee g_3(t))), \quad p_7^3 = 0.31
\end{cases} \\
g_8(t+1) = f_8^1\left(g_{10}(t)\right) = g_{10}(t), \quad p_8^1 = 1 \\
g_9(t+1) = f_9^1\left(g_{12}(t)\right) = g_{12}(t), \quad p_9^1 = 1 \\
g_{10}(t+1) = f_{10}\left(g_6(t),\, g_8(t),\, g_9(t),\, g_{11}(t),\, g_{12}(t)\right) = \\
\qquad = \begin{cases}
f_{10}^1\left(g_8(t)\right) = g_8(t), \quad p_{10}^1 = 0.36 \\
f_{10}^2\left(g_6(t)\right) = g_6(t), \quad p_{10}^2 = 0.33 \\
f_{10}^3\left(g_9(t),\, g_{11}(t),\, g_{12}(t)\right) = \\
\quad = (\neg g_9(t) \wedge (g_{11}(t) \vee g_{12}(t))) \vee \\
\quad \vee (g_9(t) \wedge g_{11}(t)), \quad p_{10}^3 = 0.31
\end{cases} \\
g_{11}(t+1) = f_{11}^1\left(g_6(t)\right) = g_6(t), \quad p_{11}^1 = 1 \\
g_{12}(t+1) = f_{12}\left(g_1(t),\, g_2(t),\, g_3(t),\, g_9(t),\, g_{10}(t),\, g_{11}(t)\right) = \\
\qquad = \begin{cases}
f_{12}^1\left(g_2(t),\, g_9(t),\, g_{11}(t)\right) = \\
\quad = (\neg g_2(t) \wedge (g_9(t) \wedge g_{11}(t))) \vee \\
\quad \vee (g_2(t) \wedge (g_9(t) \vee g_{11}(t))), \quad p_{12}^1 = 0.35 \\
f_{12}^2\left(g_1(t),\, g_3(t),\, g_{10}(t)\right) = \\
\quad = (\neg g_{10}(t) \wedge (g_1(t) \wedge \neg g_3(t)) \vee \\
\quad \vee (g_10(t) \wedge (g_1(t) \vee \neg g_3(t))), \quad p_{12}^2 = 0.33 \\
f_{12}^3\left(g_1(t)\right) = g_1(t), \quad p_{12}^3 = 0.32.
\end{cases}
\end{cases} \tag{8.1.4}$$

As done in the previous case, let us prove that the results obtained in [42] are correct by resorting to some simple simulations in Matlab.

Starting from the initial condition

$$\begin{bmatrix} G_1 & G_2 & G_3 & G_4 & G_5 & G_6 & G_7 & G_8 & G_9 & G_{10} & G_{11} & G_{12} \end{bmatrix} =$$
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

the evolution of the system after 500 iterations is shown in Fig. 8.1.7, while the evolution of the three cancer genes is shown in Fig. 8.1.8. One can observe that the three cancer genes expressions oscillate until a certain time step and then become equal to zero throughout the rest of the simulation.

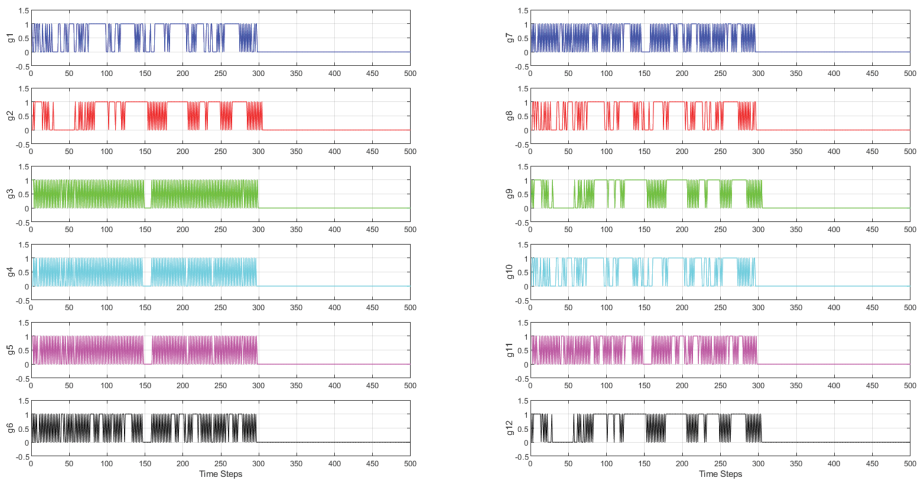

Figure 8.1.7: Time evolution of the system in non-tumor networks

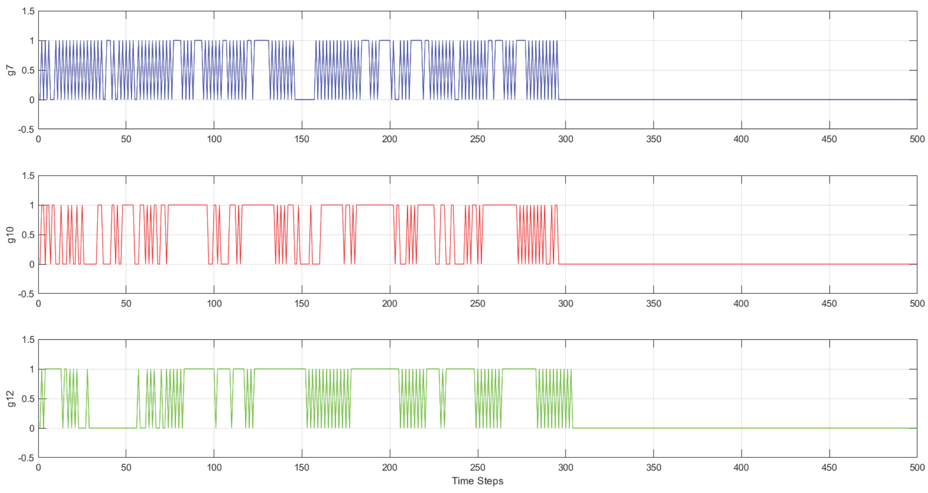

Figure 8.1.8: Time evolution of the three cancer genes in non-tumor networks

The next graph, presented in Fig. 8.1.9, is obtained by running 1000 simulations with 4000 iterations each, and random initial condition (assuming that the cancer genes are unexpressed), and by counting the number of times the three cancer states end up in each of the eight possible combinations. The graph shows that the desired combination (0, 0, 0), corresponding to the stable case where the cancer genes

$g7$, $g10$ and $g12$ are unexpressed, occurs 100% of the times, while any undesired combination, where at least one cancer gene is expressed, never occurs.

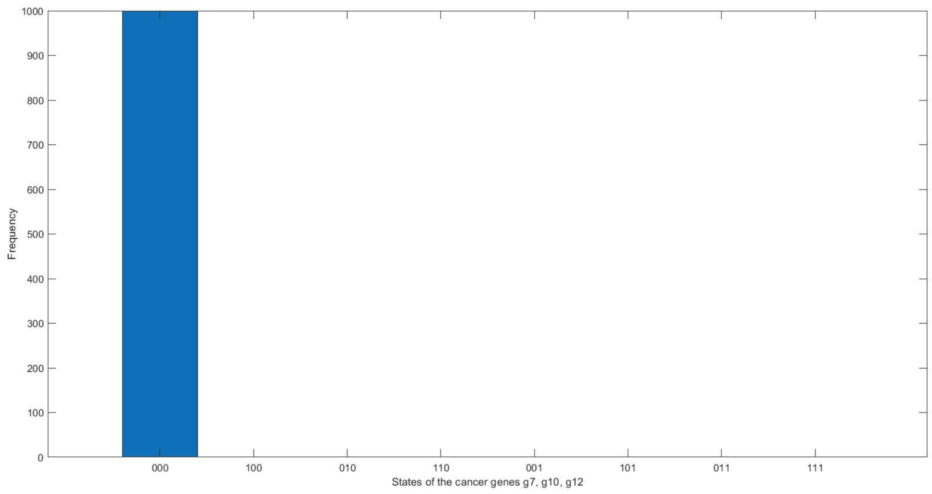

Figure 8.1.9: Steady state behaviour of the three cancer genes in non-tumor networks

In order to convert the logic system into its algebraic form, the number of networks of the P.B.N.is fundamental and, in this case, it is equal to $N = \prod_{i=1}^{12} l_i = l_1 \cdot \ldots \cdot l_{12} = 1 \cdot \ldots \cdot l_7 \cdot l_{10} \cdot l_{12} = 1 \cdot \ldots \cdot 3 \cdot 3 \cdot 3 = 27$, (see Prop. 4.1.1).
The structure matrices of the twenty-seven B.N.s are computed by using a function provided by Cheng in [7] as before.
The twenty-seven structure matrices and their corresponding probabilities are listed below:

$$L_1 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^1_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^1_{12} = 0.0466$$
$$L_2 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^2_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^2_{12} = 0.0440$$
$$L_3 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^3_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^3_{12} = 0.0426$$
$$L_4 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^1_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^1_{12} = 0.00427$$
$$L_5 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^2_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^2_{12} = 0.0403$$
$$L_6 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^3_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^3_{12} = 0.0391$$
$$L_7 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^1_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^1_{12} = 0.0401$$
$$L_8 = M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^2_{f_{12}},$$
$$\text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^2_{12} = 0.0379$$

$$
\begin{aligned}
L_9 &= M^1_{f_1} * \ldots * M^1_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^1_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^3_{12} = 0.0367\\
L_{10} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^1_{12} = 0.0403\\
L_{11} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^2_{12} = 0.0380\\
L_{12} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^3_{12} = 0.0369\\
L_{13} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^1_{12} = 0.0370\\
L_{14} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^2_{12} = 0.0348\\
L_{15} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^3_{12} = 0.0338\\
L_{16} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^1_{12} = 0.0347\\
L_{17} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^2_{12} = 0.0327\\
L_{18} &= M^1_{f_1} * \ldots * M^2_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^2_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^3_{12} = 0.0317\\
L_{19} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^1_{12} = 0.0391\\
L_{20} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^2_{12} = 0.0368\\
L_{21} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^1_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^1_{10} \cdot \ldots \cdot p^3_{12} = 0.0357\\
L_{22} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^1_{12} = 0.0358\\
L_{23} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^2_{12} = 0.0338\\
L_{24} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^2_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^2_{10} \cdot \ldots \cdot p^3_{12} = 0.0327\\
L_{25} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^1_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^1_{12} = 0.0336\\
L_{26} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^2_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^2_{12} = 0.0317\\
L_{27} &= M^1_{f_1} * \ldots * M^3_{f_7} * \ldots * M^3_{f_{10}} * \ldots * M^3_{f_{12}},\\
&\quad \text{with } P_1 = p^1_1 \cdot \ldots \cdot p^3_7 \cdot \ldots \cdot p^3_{10} \cdot \ldots \cdot p^3_{12} = 0.0308.
\end{aligned}
\tag{8.1.5}
$$

It is now possible to simulate the evolution of this P.B.N. using the algebraic form, and hence the twenty-seven structure matrices. In Figs. 8.1.10 and 8.1.11, the time evolution of the states and the time evolution of the three cancer genes are reported. Both graphs confirm the fact that the algebraic form of the P.B.N. reflects the dynamics of the logic system since the steady state δ_{4096}^{4096} is reached before the end of the simulation and consequently the three cancer genes are equal to zero.

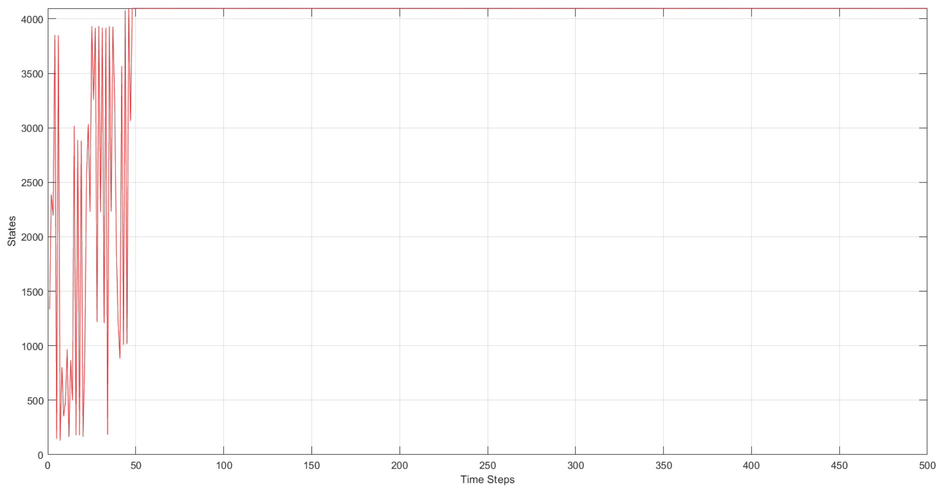

Figure 8.1.10: Time evolution of the states in non-tumor networks

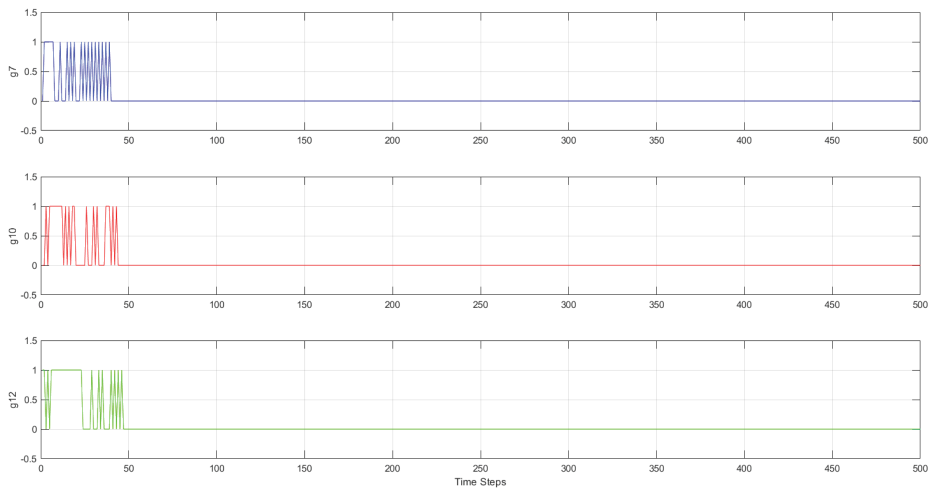

Figure 8.1.11: Time evolution of the three cancer genes in non-tumor networks

From the update equations and using Def. 4.2.3, it is possible to derive the incidence matrix as in the following:

$$\mathcal{I}(\mathbf{G}) = \bigvee_{\gamma=1}^{27} \mathcal{I}(\mathbf{G}, \Sigma_\gamma) = \begin{bmatrix} 0&0&0&0&0&0&1&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0&0&0&0&1\\ 0&0&1&0&1&0&1&0&0&0&0&0\\ 0&0&1&1&0&0&1&0&0&0&0&0\\ 0&0&0&1&1&0&1&0&0&1&1&0\\ 0&0&1&1&1&1&0&0&0&0&0&0\\ 1&0&0&0&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&0&0&0&1&0&1\\ 0&0&0&0&0&0&1&1&0&0&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&1\\ 0&1&0&0&0&1&0&0&1&1&0&0 \end{bmatrix} \tag{8.1.6}$$

From the latter, it is easy to draw the graph of this P.B.N., which is shown in Fig. 8.1.12. The three cancer genes are highlighted in different colours.

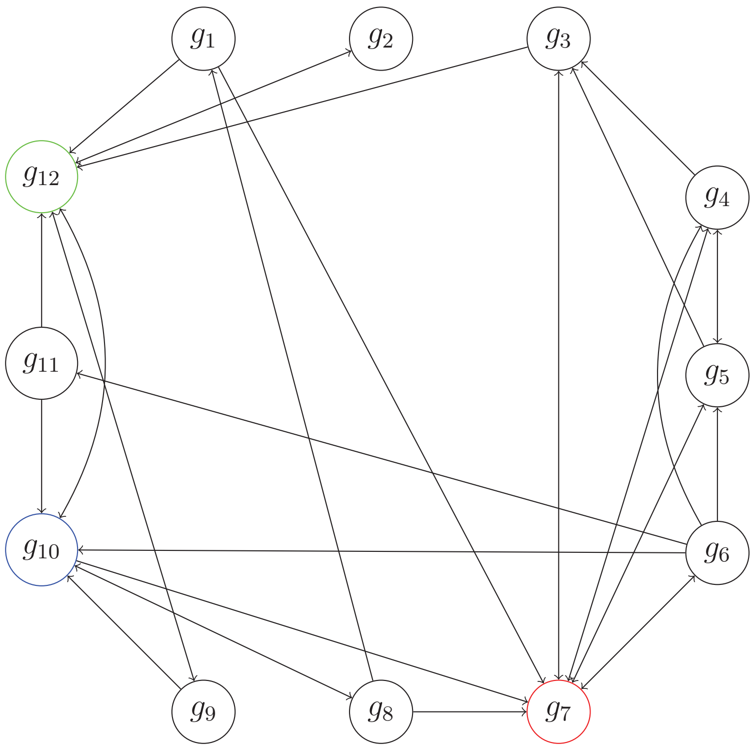

Figure 8.1.12: Graph of the Boolean Network for non-tumor cells

Instead, the state graph of this P.B.N. is too complex to be drawn because it is composed of $2^{12} = 4096$ states and $2^{12} \cdot 12 = 49152$ edges.

In the following, an analysis of the fixed points, limit cycles and stability of the P.B.N. is presented.
The number of fixed points for each of the twenty-seven B.N.s is as follows:

$$N_p = [12\ 6\ 6\ 6\ 3\ 3\ 6\ 3\ 3\ 12\ 6\ 6\ 6\ 3\ 3\ 6\ 3\ 3\ 8\ 4\ 4\ 2\ 1\ 1\ 2\ 1\ 1].$$

Among these fixed points, there is only one common fixed point (see Def. 4.3.2): $\delta_{p_c} = \delta_{4096}^{4096}$. With a simulation time long enough, the state evolution always reaches this state. Since it is a common fixed point, it remains equal to δ_{4096}^{4096} throughout the rest of the simulation. As discussed before, the other fixed points can be reached but, since they are fixed points of only one or a few structure matrices, corresponding to a particular structure matrix, there is a non-zero probability to move from them to another state. This explains an initial oscillatory behaviour of the P.B.N., while the evolution finally converges to a single state.
For what regards the limit cycles, the computational problem previously discussed arises also in this case, as the number of structure matrices is even larger.
Let us analyse the global stability with probability one of this P.B.N., (see Thm. 4.4.1). In this case, only the last matrix row is unitary. Therefore, one can claim that the state δ_{4096}^{4096} is globally stable with probability one, as observed in the previous simulations.

The references examined to write this chapter are the following: [13], [15], [41], [42].

Improvements and Conclusions

In this chapter, other interesting topics to deepen the knowledge on B.N.s are presented. Then, some conclusions are drawn.

Improvements

In the last years, there has been a growing interest in the topic of Boolean Networks due to the promising applications in representing and analysing Gene Regulatory Networks but also in other fields, such as Game Theory. Especially for the last topic, k-valued networks represent an important generalisation of B.N.s since they allow to model systems with variables that can assume k different values. The generalisation is explained in [11].
Other Systems Theory properties for B.N.s and P.B.N.s have been largely treated: observability, reconstructibility and the design of a state observer. Their analysis can be found, for instance, in [43], [44], [45] and [29]. As regards feedback control, in case the state information is not completely accessible, [46] presents an output feedback stabilisation technique.
The choice of the input sequence for a specific trajectory of a B.C.N. can be done according to an optimal control problem where, for example, the objective is to minimise a cost function. The theory is explained in [8].
Furthermore, the problems of disturbance decoupling and input/output decoupling are solved in [47] and [48]. The first problem corresponds to finding suitable controls such that for the closed-loop system the outputs are not affected by the disturbances. The second problem requires that, for every initial state, two input sequences, whose entries coincide at every time instant, generate two output sequences with equal entries for every time instant.
Another interesting problem for B.N.s and P.B.N.s is fault detection, as explained in [49] and in [28]. The problem consists in detecting the switching from a non-faulty configuration to a faulty one, both online, where the input sequence is arbitrary, and offline, where a specific input sequence is chosen.
As shown in [50], it is possible to realize output tracking of B.C.N.s by converting this problem into a specific set controllability problem. Given a reference system and any initial condition, the idea is to find a sequence of inputs such that the real input corresponds to the input of the reference system at every time instant.
A recent study in [9] presents the hidden order of B.N.s, which is determined by the fixed points and limit cycles of their dual networks. This order describes the evolution of the overall network rather than the evolution of a single state.

Conclusions

The initial part of this thesis focused on Boolean logic because it represents the basic knowledge to understand the structure of Boolean Networks. These networks are, in fact, dynamical systems described by logical variables and functions. The usefulness of a B.N. is also determined by the possibility to model a system in algebraic form by means of the left semi-tensor product, an aspect that was thoroughly explained. The algebraic form, summed up in the transition matrix L, contains the information to analyse the network both quantitatively and qualitatively. Another option to understand the dynamical behaviour of a B.N. is the network graph and more importantly the state graph, which are the graphical representations respectively in the logic and algebraic domain. Next, the asymptotic behaviour of a B.N. was investigated to analyse its convergence properties: attractors, such as fixed points and limit cycles, basins of attraction, absorption time and communicability. Then, the problem of stability was presented, differentiating global stability and globally attractive cycles.
Afterwards, the extension of B.N.s to Boolean Control Networks due to the introduction of Boolean inputs was presented. It was shown how these networks can be considered as switched systems and how the information on the different B.N.s can be summed up in the matrices L_{tot} and $\mathbb{L}$. Thanks to the latters, Systems Theory properties such as reachability and controllability are easily verifiable. The state-feedback stabilisation of this network was also considered. An algorithm to compute the feedback matrix that stabilises the system in the minimum number of steps was presented.
Subsequently, the stochastic extension of B.N.s represented by Probabilistic Boolean Networks was introduced. A P.B.N. is a collection of B.N.s whose choice depends on a probability. It was shown how to describe these networks both algebraically and graphically, and some probability insights were explained in detail. Then, the convergence properties derived from the B.N.s were adapted to the uncertainty of the new model. Also the problem of stability was restated as the problem of stability with probability one. In fact, in a probabilistic model, different types of stabilities exist: with probability one, in probability and in distribution.
The final step was to study the effect of inputs on P.B.N.s, that consequently become Probabilistic Boolean Control Networks. Also these networks can be described as switched systems. A new matrix, L_{TOT}, was introduced to summarise the information of each B.C.N.. The Systems Theory properties of reachability, controllability and feedback control were adapted to these new networks.
In order to test the validity of the theory presented, two applications to Gene Regulatory Networks were introduced: the oxidative stress response and the early detection of cancer.
In the first application, the B.C.N. model with an arbitrary input was created, starting from the logical difference equations of the system. Thanks to software, the two structure matrices were easily computed and, with the help of simulations and their related graphs, a quantitative and qualitative analyses were made.
In the second application, the P.B.N. model was more difficulty to derive due to the size of the network and the number of different structure matrices. As for the first model, a few simulations were run to test the correctness of the model with respect to the original system and an analysis of the attractors and of stability was made.

Appendices

Appendix A

Matlab Code

This chapter contains the functions used to analyse the B.N. and the P.B.N. in the two proposed applications. The functions were written in Matlab.

A.1 General Functions

Here are presented some basic functions that were useful for the simulations.

A.1.1 DeltafromNumber()

This function computes the *num*-th canonical column vector of dimension n.

```
function delta_col = DeltafromNumber(num, n)
% From value to column vector

    delta_col = zeros(2^n,1);
    delta_col(num) = 1;

end
```

A.1.2 NumberfromDelta()

This function computes the value of the canonical column vector of dimension n.

```
function [num, n] = NumberfromDelta(delta_col)
% From column vector to value

    n = log2(length(delta_col));
    num = find(delta_col==1);

end
```

A.1.3 fromBVtoLV()

This function computes the j-th value of the canonical column vector of dimension n from a set of boolean variables.

```matlab
function [j, n] = fromBVtoLV(vars)
% From boolean variables to value of delta

    n = size(vars,2);
    vars = vars(1,:);

    j = 0;
    for i=1:1:n
        j = j + (1-vars(i))*2^(n-i);
    end
    j = j + 1;

end
```

A.1.4 LfromLrow()

This function computes the transition matrix L from its row representation.

```matlab
function L = LfromLrow(Lrow, rows)
% Structure matrix L from its row version

    if ~exist('rows','var')
        rows = size(Lrow,2);
    end

    L = zeros(rows,size(Lrow,2));

    for i=1:1:size(Lrow,2)

        L(Lrow(i),i) = 1;
    end

end
```

A.1.5 transitionMatrixL()

This function computes the transition matrix L from a set of variables and a set of corresponding update equations. It uses functions from the STP Toolbox created by D. Cheng and available here: `http://lsc.amss.ac.cn/~dcheng/`.

```matlab
function [L, Lrow] = transitionMatrixL(variables, equations)
% Structure matrix L and its row version

    % Structure matrices of fundamental operators
    k = 2;
    ME = lme(k);
    MI = lmi(k);
    MD = lmd(k);
    MN = lmn(k);
    MR = lmr(k);
    MC = lmc(k);
    MU = lmu(k);
    MX = lm([2 1 1 2], 2);

    % Creating the set of variables and corresponding update ...
        equations
    options = lmset('vars',variables);
    eqn = equations;

    % Computing the transition matrix L
    eqn = completeeqn(eqn,options);
    expr = stdform(eqn,options,k);
    for i=1:length(expr)
        expr{i} = eval(expr{i});
    end
    L = ctimes(expr{:});
    Lrow = L.v;
    L = LfromLrow(Lrow,size(Lrow,1));

end
```

A.1.6 LSTP()

This function computes the left semi-tensor product between two matrices.

```
function C = LSTP(A, B)
% Left semi-tensor product

    c1 = size(A,2);
    r2 = size(B,1);

    % Matrix product if compatible dimensions
    if c1==r2
        C = A*B;
    else
        T = lcm(c1,r2);
        C = kron(A,eye(T/c1))*kron(B,eye(T/r2));
    end

end
```

A.1.7 L_power()

This function computes the p-th power of the matrix L.

```
function power = L_power(L, p)
% Power of structure matrix L

    power = L;
    if p == 1

    else
        for s=2:1:p
            power = LSTP(power,L);
        end
    end

end
```

A.1.8 properFactors()

This function computes the proper factors of a number.

```
function proper_factors = properFactors(num)
% Proper factors

    proper_factors = 1;

    div = 2;
    while div<num

        if mod(num,div)==0
            proper_factors = [proper_factors, div];
        end
        div = div + 1;
    end

end
```

A.1.9 FixedPoints()

This function computes the number and values of the fixed points.

```
function [num_fixedPoints, fixedPoints] = FixedPoints(L)
% Function to compute the number and values of the fixed points

    num_fixedPoints = trace(L);

    fixedPoints = num2cell(find(diag(L)==1));

end
```

A.1.10 AbsorptionTime_r_0()

This function computes the absorption time of a boolean network with the smallest exponent r_0.

```
function T_t = AbsorptionTime_r_0(L)
% Absorption time of the boolean network with exponent r_0

    L_powers=cell(1, size(L,1));
    L_row_powers=cell(1, size(L,1));

    % Powers of L in row form
    for p=1:1:size(L,1)
        L_powers{p}=L_power(L,p);
        L_row_powers{p}=LrowfromL(L_powers{p});
    end

    % Cycle for determining r_0
    r_0=size(L,1);
    for j=1:1:size(L,1)
        for i=1:1:size(L,1)
            if isequal(L_row_powers{j},L_row_powers{i}) && j~=i ...
                && j<r_0
                r_0=j;
            end
        end
    end
    T_t = r_0;

end
```

A.1.11 isStable()

This function verifies if the boolean network is globally stable.

```
function isStable(L)
% Check if the B.N. is globally stable

    r_0 = AbsorptionTime_r_0(L);
    L_powers=cell(1, size(L,1));
    L_row_powers=cell(1, size(L,1));
    stable_powers=zeros(1,size(L,1));

    % Powers of L in row form starting from the r_0-th power
    for p=r_0:1:size(L,1)
        L_powers{p}=L_power(L,p);
        L_row_powers{p}=LrowfromL(L_powers{p});
    end

    % Checking stability
    for p=r_0:1:size(L,1)
        if all(L_row_powers{p} == L_row_powers{r_0}(1,1))
            stable_powers(1,p)=1;
        end
    end
    if sum(stable_powers,2)==size(L,1)-r_0+1
        disp("Stable")
    else
        disp("Not stable")
    end

end
```

A.1.12 LimitCycles_modified()

This function computes the number and values of the limit cycles.

```
function [num_limitCycles, limitCycles] = ...
    LimitCycles_modified(L, max_cycle_length)
% Function to compute the number and the values of the limit cycles

    U = 1:size(L,1);
    num_limitCycles = zeros(1, size(L,1));
    num_limitCycles(1) = trace(L);
    C_s = cell(1, size(L,1));
    C_s_c = cell(1, size(L,1));
    D_s = cell(1, size(L,1));
    C_s{1} = find(diag(L)==1)';
    C_s_c{1} = setdiff(U,C_s{1});
    D_s{1} = C_s{1};

    s=2;
    while s<=max_cycle_length

        proper_factors = properFactors(s);

        sum = 0;
        for i=1:1:length(proper_factors)
            sum = sum + ...
                proper_factors(i)*num_limitCycles(proper_factors(i));
        end

        num_limitCycles(s) = (trace(L^s) - sum) / s;

        C_s{s} = find(diag(L^s)==1)';
        C_s_c{s} = setdiff(U,C_s{s});

        inter = U;
        for i=1:1:length(proper_factors)
            inter = intersect(inter,C_s_c{proper_factors(i)});
        end
        D_s{s} = intersect(C_s{s},inter);

        s=s+1;

    end
    limitCycles = D_s;

end
```

A.1.13 commonFixedPointsPBN()

This function computes the number and values of the common fixed points in a probabilistic boolean network.

```
function [num_fixedPoints, fixedPoints, common_fixedPoints] = ...
    commonFixedPointsPBN(L_cell)
% Common fixed points in a probabilistic boolean network

    % Number of fixed points and corresponding values
    num_fixedPoints = cell(1,length(L_cell));
    fixedPoints = cell(1,length(L_cell));
    for i=1:1:length(L_cell)
```

```
        [num_fixedPoints{i}, fixedPoints{i}] = ...
            FixedPoints(L_cell{i});
    end

    [minVal,minIdx] = min([num_fixedPoints{:}]);

    % Common fixed points
    common_fixedPoints = zeros(1,minVal);
    for j=1:1:minVal
        for k=1:1:length(L_cell)
            if k==minIdx
                break
            else
                for l=1:1:size(fixedPoints{k},1)
                    if fixedPoints{1,k}{1,1} == ...
                        fixedPoints{1,minIdx}{j,1}
                        common_fixedPoints(j) = ...
                            fixedPoints{1,minIdx}{j,1};
                    else
                        common_fixedPoints(j) = 0;
                    end
                end
            end
        end
    end
    common_fixedPoints = nonzeros(common_fixedPoints);

end
```

A.1.14 limitCyclesPBN_brute_force()

This function computes the number and values of the limit cycles in a probabilistic boolean network with brute force approach in order to find the maximum length of the cycles.

```
function [cycles_cell, max_cycle_length] = ...
    limitCyclesPBN_brute_force(L_cell)
% Limit cycles in a probabilistic boolean network (brute force)

    % Cycles listed in ordered columns that indicate the length
    cycles_cell = cell(1,length(L_cell));
    for l=1:1:length(L_cell)

        % Array of states
        states = 1:1:size(L_cell{1,1},1);
        nums = ones(1,size(L_cell{1,1},1));
        cycles = {};

        % Cycles of each matrix L_t
        while isempty(states)==0

            % Possible cycle with first element of states which ...
                is removed
            possible_cycle = states(1,1);
            result = NumberfromDelta(LSTP(L_cell{l}, ...
                       DeltafromNumber(states(1,1), ...
                       log2(size(L_cell{1,1},1)))));
                  states(states==states(1,1)) = [];

                  % Check if cycle of length one (fixed point)
                  if result==possible_cycle(1,1)
                      cycles{nums(1,1),1} = possible_cycle;
                      nums(1,1) = nums(1,1)+1;

                  % Add the result state in the possible cycle
                  else
                      possible_cycle = [possible_cycle,result];

                      % Determine if it is a cycle or a false cycle
                      while 1
                          result = NumberfromDelta(LSTP(L_cell{1}, ...
                              DeltafromNumber(result, ...
                              log2(size(L_cell{1,1},1)))));

                          % Check if really a cycle
                          if result==possible_cycle(1,1)
                              cycles{nums(1,length(possible_cycle)), ...
                                  length(possible_cycle)} = ...
                                  possible_cycle;
                              nums(1,length(possible_cycle)) = ...
                                  nums(1,length(possible_cycle))+1;

                              % Delete all the states in the cycle
                              for i=2:1:length(possible_cycle)
                                  states(states==possible_cycle(i)) = [];
                              end
                              break

                          % Check if the result is equal to one of the ...
                              previous states in the possible cycle
                          else

                              % If not, add it
                              if result~=possible_cycle(1,2:end)
                                  possible_cycle = ...
                                      [possible_cycle,result];

                              % If yes, not a cycle
                              else
                                  break
                              end
                          end
                      end
                  end
              end
              cycles_cell{1,1} = cycles;
          end

          % Max length of cycles in the PBN
          max_cycle_length = max(cellfun('size',cycles_edc,2));

      end
```

A.1.15 limitCyclesPBN()

This function computes the number and values of the limit cycles in a probabilistic boolean network using the knowledge on the maximum length of the cycles.

```matlab
function limitCycles_finals = limitCyclesPBN(L_cell, ...
    max_cycle_length)
% Limit cycles in a probabilistic boolean network

    % Number of limit cycles and values belonging to the cycles
    num_limitCycles = cell(1,length(L_cell));
    limitCycles = cell(1,length(L_cell));
    for i=1:1:length(L_cell)
        [num_limitCycles{i}, limitCycles{i}] = ...
            LimitCycles_modified(L_cell{i}, max_cycle_length);
    end

    % Cycle to find the values belonging to each cycle
    limitCycles_finals = cell(size(limitCycles));
    for l=1:1:length(limitCycles)
        limitCycles_final = cell(1,size(L_cell{l},1));
        numbers_in_limitCycles = limitCycles{1,l};

        % Cycle of length equal to the largest limit cycle
        for i=1:1:max_cycle_length
            number_of_cycles = length(numbers_in_limitCycles{1,i});
            cycles = cell(number_of_cycles/i,1);
            cycle = zeros(1,i);
            num=1;

            % Cycle to determine the values that compose a limit ...
                cycle
            while isempty(numbers_in_limitCycles{1,i})==0
                result = numbers_in_limitCycles{1,i}(1,1);
                cycle(1,1) = result;
                numbers_in_limitCycles{1,i} ...
                    (numbers_in_limitCycles{1,i}==result) = [];
                for j=1:1:i-1
                    result = NumberfromDelta(LSTP(L_cell{l}, ...
                        DeltafromNumber(result, ...
                        log2(size(L_cell{1,1},1)))));
                    cycle(1,j+1) = result;
                    numbers_in_limitCycles{1,i} ...
                        (numbers_in_limitCycles{1,i}==result) = [];
                end
                cycles{num,1} = cycle;
                num = num+1;
            end
            limitCycles_final{1,i} = cycles;
        end
        limitCycles_finals{1,l} = limitCycles_final;
    end

end
```

A.1.16 stabilityPBN()

This function verifies if the probabilistic boolean network is globally stable with probability one.

```
function stable_points = stabilityPBN(L_cell)
% Global stability in a probabilistic boolean network

    % Boolean sum of the twelve structure matrices
    L_sum = 0;
    for i=1:1:length(L_cell)
        L_sum = or(L_sum,L_cell{1,i});
    end

    % Boolean sum of the boolean powers of L_sum
    L_power = L_sum;
    L_sum_power = L_sum;
    for i=2:1:size(L_cell{1,1},1)
        L_power = LSTP(L_power,L_sum);
        L_power = L_power~=0;
        L_sum_power = or(L_sum_power,L_power);
    end

    % Array of states where the P.B.N. is globally stable
    stable_points = find(sum(L_sum_power,2)==size(L_cell{1,1},1));

end
```

A.2 Functions for the Oxidative Stress Response model

Here are presented the functions that were used for the simulation of the oxidative stress response model.

A.2.1 simulationOxidativeStressResponse_variables()

This function simulates the dynamics of the system variables according to the value of *Stress*.

```
function x = simulationOxidativeStressResponse_variables(x_0, ...
    stress, sim_time)
% Simulation of the oxidative stress response model

    % Initial condition
    if stress==0
        u = zeros(1,sim_time);
    elseif stress==1
        u = ones(1,sim_time);
    else
        u = zeros(1,100);
        u(1,25:74) = 1;
    end

    % Variables vector
    x = zeros(6,sim_time);
    x(:,1) = x_0;

    for t=2:1:sim_time

        % Variables update
        x(1,t) = u(t-1)&(~x(6,t-1));
        x(2,t) = ~x(1,t-1)&(x(4,t-1)|x(2,t-1));
        x(3,t) = x(1,t-1)&(~x(6,t-1));
        x(4,t) = x(3,t-1)|(~x(2,t-1));
        x(5,t) = ~x(1,t-1);
        x(6,t) = x(4,t-1)&((~x(6,t-1))|(~x(5,t-1)));
    end

    % Graph

end
```

A.2.2 simulationOxidativeStressResponse_Lmatrix()

This function simulates the dynamics of the system by means of the transition matrix L and according to the value of *Stress*.

```
function x = simulationOxidativeStressResponse_Lmatrix(x_0, ...
    stress, L, sim_time)
% Simulation of the oxidative stress response model

    % Input
    if stress==0
        u = zeros(1,sim_time);
    elseif stress==1
        u = ones(1,sim_time);
    else
        u = zeros(1,100);
        u(1,25:74) = 1;
    end

    x = zeros(size(L,1),sim_time);

    % Initial condition into logic vector
    j=fromBVtoLV(x_0');
    x(:,1)=LfromLrow(j,size(L,1));

    for t=2:1:sim_time

        % Choice of partition of L and states update
        if u(t-1)==0
            x(:,t)=LSTP(L(:,size(L,2)/2+1:size(L,2)),x(:,t-1));
        else
            x(:,t)=LSTP(L(:,1:size(L,2)/2),x(:,t-1));
        end
    end

    x_row=LrowfromL(x);
```

```matlab

    % Graph

end
```

A.3 Functions for the Early Detection of Cancer model

Here are presented the functions that were used for the simulation of the early detection of cancer model.

A.3.1 simulationEarlyDetectionCancer_variables()

This function simulates the dynamics of the system variables according to the value of *Tumor*.

```matlab
function x = simulationEarlyDetectionCancer_variables(x_0, ...
    tumor, sim_time, sim_graphs)
% Simulation of the early detection of cancer model

    % Variables vector
    x = zeros(12,sim_time);
    x(:,1) = x_0;

    if tumor==1

        % Probabilities for update functions for Tumor Cells
        p_4_1=0.58;
        p_4_2=0.42;

        p_10_1=0.34;
        p_10_2=0.33;
        p_10_3=1-p_10_1-p_10_2;

        p_12_1=0.55;
        p_12_2=1-p_12_1;

        for t=2:1:sim_time

            % Random number generation from uniform distribution ...
                [0,1]
            num_4 = rand();
            num_10 = rand();
            num_12 = rand();

            % Variables update
            x(1,t) = (~x(5,t-1)&(~x(6,t-1)&x(12,t-1)))| ...
                (x(5,t-1)&(~x(6,t-1)|x(12,t-1)));
            x(2,t) = x(11,t-1);
            x(3,t) = (~x(8,t-1)&(x(4,t-1)&x(9,t-1)))| ...
                (x(8,t-1)&(x(4,t-1)|x(9,t-1)));
            if num_4<p_4_1
                x(4,t) = x(8,t-1);
            else
                x(4,t) = x(10,t-1);
            end
            x(5,t) = ~x(6,t-1)&(~x(12,t-1)&x(1,t-1))|x(6,t-1);
            x(6,t) = (~x(12,t-1)&(~x(1,t-1)&x(5,t-1)))| ...
                (x(12,t-1)&x(5,t-1));
            x(7,t) = x(10,t-1);
            x(8,t) = x(4,t-1);
            x(9,t) = x(3,t-1);
            if num_10<p_10_1
                x(10,t) = x(7,t-1);
            elseif num_10>=p_10_1 && num_10<p_10_1+p_10_2
                x(10,t) = (~x(1,t-1)&(x(5,t-1)|x(6,t-1)))| ...
                    (x(1,t-1)&(x(5,t-1)&x(6,t-1)));
            else
                x(10,t) = (~x(5,t-1)&(x(6,t-1)&~x(12,t-1)))| ...
                    (x(5,t-1)&(x(6,t-1)|~x(12,t-1)));
            end
            x(11,t) = x(2,t-1);
            if num_12<p_12_1
                x(12,t) = (~x(1,t-1)&(~x(5,t-1)&x(6,t-1)))| ...
                    (x(1,t-1)&(~x(5,t-1)|x(6,t-1)));
            else
                x(12,t) = x(2,t-1);
            end
        end

        if sim_graphs==1

            % Graphs

        end

    else

        % Probabilities for update functions for Non-tumor Cells
        pp_7_1=0.37;
        pp_7_2=0.32;
        pp_7_3=1-pp_7_1-pp_7_2;

        pp_10_1=0.36;
        pp_10_2=0.33;
        pp_10_3=1-pp_10_1-pp_10_2;

        pp_12_1=0.35;
        pp_12_2=0.33;
        pp_12_3=1-pp_12_1-pp_12_2;

        for t=2:1:sim_time

            % Random number generation from uniform distribution ...
                [0,1]
            num_7=rand();
            num_10=rand();
            num_12=rand();

            % Variables update
            x(1,t)=x(8,t-1);
```

```
                x(2,t)=x(12,t-1);
                x(3,t)=(~x(7,t-1)&(~x(4,t-1)&x(5,t-1)))| ...
                    (x(7,t-1)&(~x(4,t-1)|x(5,t-1)));
                x(4,t)=(~x(5,t-1)&(~x(6,t-1)&x(7,t-1)))| ...
                    (x(7,t-1)&x(5,t-1));
                x(5,t)=(~x(6,t-1)&(~x(7,t-1)&x(4,t-1)))| ...
                    (x(6,t-1)&(~x(7,t-1)|x(4,t-1)));
                x(6,t)=x(7,t-1);
                if num_7<pp_7_1
                    x(7,t)=(~x(4,t-1)&x(6,t-1))|(x(4,t-1)& ...
                        (~x(5,t-1)|x(6,t-1)));
                elseif num_7>=pp_7_1 && num_7<pp_7_1+pp_7_2
                    x(7,t)=(~x(3,t-1)&(~x(8,t-1)&x(10,t-1)))| ...
                        (x(3,t-1)&(~x(8,t-1)|x(10,t-1)));
                else
                    x(7,t)=(~x(1,t-1)&x(3,t-1))| ...
                        (x(1,t-1)&(~x(8,t-1)|x(3,t-1)));
                end
                x(8,t)=x(10,t-1);
                x(9,t)=x(12,t-1);
                if num_10<pp_10_1
                    x(10,t)=x(8,t-1);
                elseif num_10>=pp_10_1 && num_10<pp_10_1+pp_10_2
                    x(10,t)=x(6,t-1);
                else
                    x(10,t)=(~x(9,t-1)&(x(11,t-1)|x(12,t-1)))| ...
                        (x(9,t-1)&x(11,t-1));
                end
                x(11,t)=x(6,t-1);
                if num_12<pp_12_1
                    x(12,t)=(~x(2,t-1)&(x(9,t-1)&x(11,t-1)))| ...
                        (x(2,t-1)&(x(9,t-1)|x(11,t-1)));
                elseif num_12>=pp_12_1 && num_12<pp_12_1+pp_12_2
                    x(12,t)=(~x(10,t-1)&(x(1,t-1))&~(x(3,t-1)))| ...
                        (x(10,t-1)&(x(1,t-1)|~(x(3,t-1))));
                else
                    x(12,t)=x(1,t-1);
                end
        end

        if sim_graphs==1

            % Graphs

        end

    end

end
```

A.3.2 simulationEarlyDetectionCancer_histogram()

This function computes the histogram of the combinations of the three cancer genes running sim_num simulations.

```matlab
function x = simulationEarlyDetectionCancer_histogram(tumor, ...
    sim_time, sim_num, sim_graphs)
% Simulation of the early detection of cancer model

    % Cycle for sim_num simulations
    hist = zeros(1,8);
    for n=1:1:sim_num

        % Random initial conditions where the cancer genes are ...
            equal to 0
        while 1
            randomNumbers = int32(randi([0, 1], [1, 12]));
            if randomNumbers(1,7)~=1 && randomNumbers(1,10)~=1 ...
                && randomNumbers(1,12)~=1
                break
            end
        end

        x_0 = randomNumbers;

        x = simulationEarlyDetectionCancer_variables(x_0, tumor, ...
            sim_time, sim_graphs);

        % Histogram array
        if tumor==1

            if x(4,sim_time)==0 && x(10,sim_time)==0 && ...
                x(12,sim_time)==0
                hist(1,1) = hist(1,1)+1;
            elseif x(4,sim_time)==1 && x(10,sim_time)==0 && ...
                x(12,sim_time)==0
                hist(1,2) = hist(1,2)+1;
            elseif x(4,sim_time)==0 && x(10,sim_time)==1 && ...
                x(12,sim_time)==0
                hist(1,3) = hist(1,3)+1;
            elseif x(4,sim_time)==1 && x(10,sim_time)==1 && ...
                x(12,sim_time)==0
                hist(1,4) = hist(1,4)+1;
            elseif x(4,sim_time)==0 && x(10,sim_time)==0 && ...
                x(12,sim_time)==1
                hist(1,5) = hist(1,5)+1;
            elseif x(4,sim_time)==1 && x(10,sim_time)==0 && ...
                x(12,sim_time)==1
                hist(1,6) = hist(1,6)+1;
            elseif x(4,sim_time)==0 && x(10,sim_time)==1 && ...
                x(12,sim_time)==1
                hist(1,7) = hist(1,7)+1;
            elseif x(4,sim_time)==1 && x(10,sim_time)==1 && ...
                x(12,sim_time)==1
                hist(1,8) = hist(1,8)+1;
            end

        else

            if x(7,sim_time)==0 && x(10,sim_time)==0 && ...
                x(12,sim_time)==0
                hist(1,1) = hist(1,1)+1;
            elseif x(7,sim_time)==1 && x(10,sim_time)==0 && ...
```

```
                    x(12,sim_time)==0
                     hist(1,2) = hist(1,2)+1;
                elseif x(7,sim_time)==0 && x(10,sim_time)==1 && ...
                    x(12,sim_time)==0
                     hist(1,3) = hist(1,3)+1;
                elseif x(7,sim_time)==1 && x(10,sim_time)==1 && ...
                    x(12,sim_time)==0
                     hist(1,4) = hist(1,4)+1;
                elseif x(7,sim_time)==0 && x(10,sim_time)==0 && ...
                    x(12,sim_time)==1
                     hist(1,5) = hist(1,5)+1;
                elseif x(7,sim_time)==1 && x(10,sim_time)==0 && ...
                    x(12,sim_time)==1
                     hist(1,6) = hist(1,6)+1;
                elseif x(7,sim_time)==0 && x(10,sim_time)==1 && ...
                    x(12,sim_time)==1
                     hist(1,7) = hist(1,7)+1;
                elseif x(7,sim_time)==1 && x(10,sim_time)==1 && ...
                    x(12,sim_time)==1
                     hist(1,8) = hist(1,8)+1;
                end

            end

        end

        % Histogram graph
        if tumor==1

            % Graph

        else

            % Graph

        end

    end
```

A.3.3 simulationEarlyDetectionCancer_Lmatrix()

This function simulates the dynamics of the system by means of the transition matrix L and according to the value of $Tumor$.

```
function x = simulationEarlyDetectionCancer_Lmatrix(x_0, tumor, ...
    L_cell, p_L_cell, sim_time)
% Simulation of the early detection of cancer model

    x=zeros(size(L_cell{1,1},1),sim_time);

    % Initial condition into logic vector
    j=fromBVtoLV(x_0');
    x(:,1)=LfromLrow(j,size(L_cell{1,1},1));

    if tumor==1

```

```
        for t=2:1:sim_time

            % Random number generation from uniform distribution ...
                [0,1]
            num=rand();

            % Selection of the current structure matrix L
            if num<p_L_cell(1,1)
                L_curr=L_cell{1};
            elseif num>=p_L_cell(1,1) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)
                L_curr=L_cell{2};
            elseif num>=p_L_cell(1,1)+p_L_cell(1,2) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)
                L_curr=L_cell{3};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3) ...
                    && num<p_L_cell(1,1)+p_L_cell(1,2)+ ...
                    p_L_cell(1,3)+p_L_cell(1,4)
                L_curr=L_cell{4};
            elseif num>=p_L_cell(1,1)+p_L_cell(1,2)+ ...
                    p_L_cell(1,3)+p_L_cell(1,4) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)
                L_curr=L_cell{5};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)
                L_curr=L_cell{6};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+ ...
                    p_L_cell(1,6)+p_L_cell(1,7)
                L_curr=L_cell{7};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+ ...
                    p_L_cell(1,6)+p_L_cell(1,7) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+ ...
                    p_L_cell(1,6)+p_L_cell(1,7)+p_L_cell(1,8)
                L_curr=L_cell{8};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                    p_L_cell(1,7)+p_L_cell(1,8) && ...
                    num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                    p_L_cell(1,7)+p_L_cell(1,8)+p_L_cell(1,9)
                L_curr=L_cell{9};
            elseif ...
                    num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                    p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                    p_L_cell(1,7)+p_L_cell(1,8)+p_L_cell(1,9) && ...
```

```
                  num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                  p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                  p_L_cell(1,7)+p_L_cell(1,8)+ ...
                  p_L_cell(1,9)+p_L_cell(1,10)
                   L_curr=L_cell{10};
              elseif ...
                  num>=p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                  p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                  p_L_cell(1,7)+p_L_cell(1,8)+ ...
                  p_L_cell(1,9)+p_L_cell(1,10) && ...
                  num<p_L_cell(1,1)+p_L_cell(1,2)+p_L_cell(1,3)+ ...
                  p_L_cell(1,4)+p_L_cell(1,5)+p_L_cell(1,6)+ ...
                  p_L_cell(1,7)+p_L_cell(1,8)+ ...
                  p_L_cell(1,9)+p_L_cell(1,10)+p_L_cell(1,11)
                   L_curr=L_cell{11};
              else
                   L_curr=L_cell{12};
              end

              % State update
              x(:,t)=LSTP(L_curr,x(:,t-1));
          end

          x_row=LrowfromL(x);
          x_row=x_row(1,1:sim_time);

          % From states to cancer genes boolean variables
          boolean_variables_12=zeros(12,sim_time);
          cancer_genes=zeros(3,sim_time);
          for i=1:1:sim_time
                  support=fromLVtoBV(x_row(1,i),12);
                  boolean_variables_12(:,i)=support(1,:)';
                  cancer_genes(1,i)=boolean_variables_12(4,i);
                  cancer_genes(2,i)=boolean_variables_12(10,i);
                  cancer_genes(3,i)=boolean_variables_12(12,i);
          end

          % Graphs

      else

          for t=2:1:sim_time

              % Random number generation from uniform distribution ...
                  [0,1]
              num=rand();

              % Selection of the current structure matrix L

                  % Similar to what has been done for tumor==1 but ...
                      with 27 matrices

              % State update
              x(:,t)=LSTP(L_curr,x(:,t-1));
          end

          x_row=LrowfromL(x);
          x_row=x_row(1,1:sim_time);
```

```

        % From states to cancer genes boolean variables
        boolean_variables_12=zeros(12,sim_time);
        cancer_genes=zeros(3,sim_time);
        for i=1:1:sim_time
                support=fromLVtoBV(x_row(1,i),12);
                boolean_variables_12(:,i)=support(1,:)';
                cancer_genes(1,i)=boolean_variables_12(7,i);
                cancer_genes(2,i)=boolean_variables_12(10,i);
                cancer_genes(3,i)=boolean_variables_12(12,i);
        end

        % Graphs

    end

end
```

Acknowledgements

Inizio con il ringraziare la Professoressa Maria Elena Valcher che è sempre stata molto disponibile in questo periodo di tesi.

Ringrazio i miei genitori Paola e Maurizio e mio fratello Matteo che mi hanno sostenuto in questi anni, consigliandomi quando ne avevo maggiore bisogno.

Ringrazio non di meno i miei amici con i quali ho condiviso le gioie e le difficoltà di questi anni universitari.

Grazie a tutti.

Bibliography

[1] Stuart A Kauffman. “Metabolic stability and epigenesis in randomly constructed genetic nets”. In: *Journal of theoretical biology* 22.3 (1969), pp. 437–467.

[2] Leshi Chen, Don Kulasiri, and Sandhya Samarasinghe. “A novel data-driven boolean model for genetic regulatory networks”. In: *Frontiers in physiology* 9 (2018), p. 1328.

[3] Daizhan Cheng. “Semi-tensor product of matrices and its application to Morgen’s problem”. In: *Science in China Series: Information Sciences* 44.3 (2001), pp. 195–212.

[4] Daizhan Cheng. “Input-state approach to Boolean networks”. In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 512–521.

[5] Tatsuya Akutsu et al. “Control of Boolean networks: Hardness results and algorithms for tree structured networks”. In: *Journal of theoretical biology* 244.4 (2007), pp. 670–679.

[6] Ilya Shmulevich et al. “Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks”. In: *Bioinformatics* 18.2 (2002), pp. 261–274.

[7] Daizhan Cheng and Hongsheng Qi. “State–space analysis of Boolean networks”. In: *IEEE Transactions on Neural Networks* 21.4 (2010), pp. 584–594.

[8] Ettore Fornasini and Maria Elena Valcher. “Optimal control of Boolean control networks”. In: *IEEE Transactions on Automatic Control* 59.5 (2013), pp. 1258–1270.

[9] Xiao Zhang, Zhengping Ji, and Daizhan Cheng. “Hidden Order of Boolean Networks”. In: *arXiv preprint arXiv:2111.12988* (2021).

[10] Daizhan Cheng and Hongsheng Qi. “Matrix expression of logic and fuzzy control”. In: *Proceedings of the 44th IEEE Conference on Decision and Control.* IEEE. 2005, pp. 3273–3278.

[11] Daizhan Cheng. “On logic-based intelligent systems”. In: *2005 International Conference on Control and Automation.* Vol. 1. IEEE. 2005, pp. 71–76.

[12] Daizhan Cheng, Hongsheng Qi, and Zhiqiang Li. *Analysis and control of Boolean networks: a semi-tensor product approach.* Springer Science & Business Media, 2010.

[13] Riccardo Sterbizzi. *Analisi e controllo delle Boolean control networks.* 2012.

[14] Nicola Sorarù. *Boolean control networks e loro applicazioni alla Teoria dei giochi*. 2019.

[15] Luca Girardi. *Boolean Control Networks: Modeling and Applications*. 2020.

[16] Ettore Fornasini and Giovanni Marchesini. *Appunti di teoria dei sistemi*. Edizioni Libreria Progetto Padova, 2011.

[17] Hongsheng Qi and Daizhan Cheng. "Analysis and control of Boolean networks: A semi-tensor product approach". In: *2009 7th Asian Control Conference*. IEEE. 2009, pp. 1352–1356.

[18] Daizhan Cheng and Hongsheng Qi. "A linear representation of dynamics of Boolean networks". In: *IEEE Transactions on Automatic Control* 55.10 (2010), pp. 2251–2258.

[19] Ettore Fornasini and Maria Elena Valcher. "Recent developments in Boolean networks control". In: *Journal of Control and Decision* 3.1 (2016), pp. 1–18.

[20] Daizhan Cheng and Hongsheng Qi. "Controllability and observability of Boolean control networks". In: *Automatica* 45.7 (2009), pp. 1659–1667.

[21] Dmitriy Laschov and Michael Margaliot. "Minimum-time control of Boolean networks". In: *SIAM Journal on Control and Optimization* 51.4 (2013), pp. 2869–2892.

[22] Ettore Fornasini and Maria Elena Valcher. "On the periodic trajectories of Boolean control networks". In: *Automatica* 49.5 (2013), pp. 1506–1509.

[23] Zdzislaw Brzezniak and Tomasz Zastawniak. *Basic stochastic processes: a course through exercises*. Springer Science & Business Media, 2000.

[24] Marcel Brun, Edward R Dougherty, and Ilya Shmulevich. "Steady-state probabilities for attractors in probabilistic Boolean networks". In: *Signal Processing* 85.10 (2005), pp. 1993–2013.

[25] Ilya Shmulevich and Edward R Dougherty. *Probabilistic Boolean networks: the modeling and control of gene regulatory networks*. SIAM, 2010.

[26] Yin Zhao and DaiZhan Cheng. "On controllability and stabilizability of probabilistic Boolean control networks". In: *Science China Information Sciences* 57.1 (2014), pp. 1–14.

[27] Yang Liu et al. "Controllability of probabilistic Boolean control networks based on transition probability matrices". In: *Automatica* 52 (2015), pp. 340–345.

[28] Thomas Leifeld, Zhihua Zhang, and Ping Zhang. "Fault detection for probabilistic Boolean networks". In: *2016 European Control Conference (ECC)*. IEEE. 2016, pp. 740–745.

[29] Ettore Fornasini and Maria Elena Valcher. "Observability and reconstructibility of probabilistic Boolean networks". In: *IEEE Control Systems Letters* 4.2 (2019), pp. 319–324.

[30] Chi Huang et al. "Stability and stabilization in probability of probabilistic Boolean networks". In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2020), pp. 241–251.

[31] Rui Li, Meng Yang, and Tianguang Chu. "State feedback stabilization for probabilistic Boolean networks". In: *Automatica* 50.4 (2014), pp. 1272–1278.

[32] Carlos Espinosa-Soto, Pablo Padilla-Longoria, and Elena R Alvarez-Buylla. "A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles". In: *The Plant Cell* 16.11 (2004), pp. 2923–2939.

[33] Fangting Li et al. "The yeast cell-cycle network is robustly designed". In: *Proceedings of the National Academy of Sciences* 101.14 (2004), pp. 4781–4786.

[34] Simone Gupta et al. "Boolean network analysis of a neurotransmitter signaling pathway". In: *Journal of theoretical biology* 244.3 (2007), pp. 463–469.

[35] Guy Karlebach and Ron Shamir. "Modelling and analysis of gene regulatory networks". In: *Nature reviews Molecular cell biology* 9.10 (2008), pp. 770–780.

[36] Nadia S Taou, David W Corne, and Michael A Lones. "Investigating the use of Boolean networks for the control of gene regulatory networks". In: *Journal of computational science* 26 (2018), pp. 147–156.

[37] Sriram Sridharan et al. "Boolean network model of oxidative stress response pathways". In: *2012 American Control Conference (ACC)*. IEEE. 2012, pp. 2737–2742.

[38] Gisela Storz and James A Imlayt. "Oxidative stress". In: *Current opinion in microbiology* 2.2 (1999), pp. 188–194.

[39] Thomas W Kensler, Nobunao Wakabayashi, Shyam Biswal, et al. "Cell survival responses to environmental stresses via the Keap1-Nrf2-ARE pathway". In: *Annual review of pharmacology and toxicology* 47.1 (2007), pp. 89–116.

[40] Jong-Min Lee and Jeffrey A Johnson. "An important role of Nrf2-ARE pathway in the cellular defense mechanism". In: *BMB Reports* 37.2 (2004), pp. 139–143.

[41] Yukinori Kurokawa et al. "Central genetic alterations common to all HCV-positive, HBV-positive and non-B, non-C hepatocellular carcinoma: a new approach to identify novel tumor markers". In: *International journal of oncology* 28.2 (2006), pp. 383–391.

[42] Danh Cong Nguyen and Farhad Azadivar. "Early detection of cancer by regression analysis and computer simulation of gene regulatory rules". In: *Journal of Medical Imaging and Health Informatics* 2.3 (2012), pp. 336–342.

[43] Ettore Fornasini and Maria Elena Valcher. "Observability and reconstructibility of Boolean control networks". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE. 2012, pp. 2574–2580.

[44] Ettore Fornasini and Maria Elena Valcher. "Observability, reconstructibility and state observers of Boolean control networks". In: *IEEE Transactions on Automatic Control* 58.6 (2012), pp. 1390–1401.

[45] Daizhan Cheng, Changxi Li, and Fenghua He. "Observability of Boolean networks via set controllability approach". In: *Systems & Control Letters* 115 (2018), pp. 22–25.

[46] Nicoletta Bof, Ettore Fornasini, and Maria Elena Valcher. "Output feedback stabilization of Boolean control networks". In: *Automatica* 57 (2015), pp. 21–28.

[47] Daizhan Cheng. “Disturbance decoupling of Boolean control networks”. In: *IEEE Transactions on Automatic Control* 56.1 (2010), pp. 2–10.

[48] Maria Elena Valcher. “Input/output decoupling of Boolean control networks”. In: *IET Control Theory & Applications* 11.13 (2017), pp. 2081–2088.

[49] Fornasini Ettore and Valcher Maria Elena. “Fault detection problems for Boolean networks and Boolean control networks”. In: *2015 34th Chinese Control Conference (CCC)*. IEEE. 2015, pp. 1–8.

[50] Xiao Zhang, Yuanhua Wang, and Daizhan Cheng. “Output tracking of Boolean control networks”. In: *IEEE Transactions on Automatic Control* 65.6 (2019), pp. 2730–2735.