

# Implicit Methods for Modeling Gene Regulatory Networks

Abhishek Garg

Submitted for the degree of Docteur Ès Sciences  
École Polytechnique Fédérale de Lausanne

July 2009



# Abstract

---

Advancements in high-throughput technologies to measure increasingly complex biological phenomena at the genomic level are rapidly changing the face of biological research from single-gene single-protein experimental approach to studying the behaviour of a gene in the context of the entire genome (and proteome). This shift in research methodologies has resulted in a new field of network biology that deals with modeling cellular behaviour in terms of network structures that represent the influence of different biological entities such as genes, proteins and metabolites on each other. These different biological entities interact with each other giving rise to a dynamical system. Even though there exists a mature field of dynamical systems theory to model such network structures, some technical challenges that are unique to biology such as the inability to measure precise kinetic information on gene-gene or gene-protein interactions and the need to model large networks comprising of thousands of nodes have renewed interest in developing new computational techniques for modeling these complex biological systems.

In this thesis, I introduce a framework for modeling such regulatory networks in biology based on Boolean algebra and finite-state machines that are reminiscent of the approach used for digital circuit synthesis and simulations in the field of *very-large-scale integration* (VLSI). The proposed formalism enables a common mathematical framework to develop computational techniques for modeling different aspects of the regulatory networks such as steady state behaviour, stochasticity and gene perturbation experiments. Further, the proposed algorithms have been implemented under the modeling toolbox **genYsis** using implicit representation techniques based on *reduced ordered binary decision diagrams* (ROBDDs) and *algebraic decision diagrams* (ADDs) enabling the modeling of large regulatory networks.

**Keywords:** Gene regulatory networks, Signaling pathways, Boolean

networks, BDDs, ADDs, Synchronous, Asynchronous, Probabilistic Boolean networks, Stochasticity, Cell differentiation, Cancer pathways, T-helper network.

# Résumé

---

Le développement de techniques avancées pour observer et mesurer l'activité de phénomènes biologiques complexes relatifs au génome ont accéléré la recherche en biologie du stade des études restreintes aux gènes ou protéines individuels, à l'étude de gènes et protéines dans un contexte plus large de génome et protéome. Cette évolution des méthodes de recherche a donné naissance à un nouveau domaine de biologie de réseau basé sur une description du comportement de la cellule en tant que réseau dans lequel plusieurs éléments interagissent, tels que gènes et protéines; une interaction qui peut être comprise dans le cadre d'un système dynamique. Bien qu'une théorie des systèmes dynamiques existe déjà, des difficultés techniques y relatives et spécifiques aux systèmes biologiques subsistent encore: par exemple, il n'est techniquement pas possible de mesurer la cinétique d'une interaction entre deux gènes ou entre un gène et une protéine; et il est en outre difficile de manipuler mathématiquement des systèmes comportant des milliers de nœuds. Ces difficultés ont relancé la nécessité d'élaborer de nouvelles méthodes de calcul pour modéliser les systèmes biologiques complexes.

Dans cette thèse, j'introduirai une base de modélisation de tels systèmes biologiques régulés en utilisant l'algèbre de Boole et les automates finis, une approche qui s'apparente à celle utilisée dans la synthèse des circuits logiques ainsi que leur simulation dans le cadre de l'intégration à très grande échelle des circuits intégrés (VLSI). Le formalisme proposé offre un cadre mathématique des techniques de calcul pour modéliser les différents aspects du réseau régulé, comme l'état stationnaire, l'aléa et les expériences sur la perturbation génétique. En outre, les algorithmes proposés ont été implémentés dans un outil de modélisation **genYsis** qui utilise des méthodes de représentation implicite basées sur les diagrammes de décision binaire réduits et ordonnés (ROBDDs) et les diagrammes de décision algébrique (ADDs), permettant ainsi de modéliser des systèmes très larges.

**Mots-clés:** Réseau de régulation génétique, Voies de signalisation, Réseaux booléens, BDDs, ADDs, Synchrones, Asynchrones, Réseau booléen probabiliste, Système stochastique, Cytodifférenciation, Voies de cancer, Réseaux de T-helper.

# Acknowledgements

---

I would like to start by thanking my advisor, Prof. Giovanni De Micheli who gave me the opportunity to pursue research in the multi-disciplinary field. I am indebted to him for his guidance and patience in the last four years. I am grateful to him for teaching me lots of things that not only led my thesis to fruition but also inculcated good researcher traits in me.

I am also thankful to Dr. Ioannis Xenarios who introduced me to the topic of modeling in biology. This thesis was possible in its current form all because of a chance interaction with Ioannis four years ago that culminated into an everlasting research collaboration. I am grateful for his patient listening to the wild and crazy ideas of a computer scientist with absolutely little background in biology.

I am also grateful to my PhD jury members - Prof. Luca Benini, Prof. Paolo Ienne and Prof. Alcherio Martinoli - for reading and evaluating my thesis in detail and for providing constructive feedback.

I would also like to thank Prof. Luis Mendoza for his guidance on my research topic during the early years of my PhD. It was also my pleasure to have worked with Dr. Alessandro Di Cara who was always ready to discuss new ideas with me and was kind enough to host me at Merck Serono for three months during my PhD. I am also grateful to Prof. Edoardo Charbon for not only his guidance during my pre-doctoral studies at EPFL but also playing the role of a mentor to whom I could always turn-up for advice. I am also thankful to Prof. Kartik Mohanram for the long research discussions, especially towards the end of my PhD, that often led to some fresh research ideas.

I am also delighted to have shared office space with Haykel in the last four years. Apart from his great company, I had the good fortune of making use of his language and negotiation skills that have been very helpful to me in dealing with various issues and making my life a lot easier in the French-speaking part of Switzerland.

I would also like to thank LSI and ESL group members for making the years that I spent at LSI a great experience. I would like to thank Srini, Shashi, Antonio, Federico, Vasilis, Ciprian, Francesco, Nicolas, Cristina, Andrea, Sandro, Nadia and David for all the great conversations and dinner parties. Many thanks to Fran, Pablo and Jaime for bringing much needed fun to LSI with

their visits. I would like to thank Rodolphe for his kindness in resolving the many system administration issues that would crop up from time to time. Special thanks must go to Chantal and Christina who were helpful beyond words in addressing administrative issues in all my years at EPFL.

Special thanks to all the friends that I made during my years in Lausanne - Satish, Shrinivas, Dinkar, Sanket, Vishwambhar, Ajay, Debasree, Gaurav, Vasu, Surender, Florian, Florence, Thomas and many more - for the wonderful Indian dinner parties, cricket matches, intellectual conversations and for all the time they always seemed to have for me.

I am also grateful to my family for providing me with constant moral support while I was away from home all these years and for believing in me.



# Contents

---

<b>Abstract</b>	<b>i</b>
<b>Résumé</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>x</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Contributions . . . . .	3
1.3 Assumptions and limitations . . . . .	4
1.4 Thesis organisation . . . . .	4
<b>2 Background</b>	<b>7</b>
2.1 Biological principles . . . . .	7
2.1.1 Cellular organisation . . . . .	7
2.1.2 Cellular signaling . . . . .	9
2.1.3 Gene regulatory networks . . . . .	10
2.2 Modeling techniques . . . . .	11
2.2.1 Continuous modeling . . . . .	11
2.2.2 Discrete modeling . . . . .	13
2.2.3 Model complexity vs. System complexity . . . . .	14
2.2.4 Quantitative vs. Qualitative . . . . .	15
2.3 Non-determinism in GRNs . . . . .	15
2.3.1 Modeling stochasticity in continuous models . . . . .	15
2.3.2 Modeling stochasticity in discrete models . . . . .	17
2.4 Boolean algebra and implicit methods . . . . .	18
2.4.1 Boolean algebra . . . . .	18
2.4.2 Finite state machines . . . . .	19
2.4.3 Binary decision diagrams . . . . .	20

2.4.4	Reachability analysis . . . . .	22
2.4.5	Discrete functions and ADDs . . . . .	23
2.5	Previous work . . . . .	24
2.5.1	Explicit enumeration based approaches . . . . .	25
2.5.2	Implicit enumeration based approaches . . . . .	28
2.5.3	Petri Nets . . . . .	28
2.5.4	Alternate formulation of GRNs in this thesis . . . . .	29
<b>3</b>	<b>Boolean Modeling</b>	<b>31</b>
3.1	Boolean mapping of GRNs . . . . .	31
3.1.1	Network states and steady states . . . . .	33
3.1.2	Synchronous vs. Asynchronous: . . . . .	34
3.2	Problem formulation . . . . .	35
3.2.1	Synchronous model . . . . .	36
3.2.2	Asynchronous model . . . . .	37
3.2.3	Boolean attractors . . . . .	39
3.2.4	Algorithms for computing attractors . . . . .	42
3.2.5	Algorithm complexity . . . . .	44
3.2.6	Computational results . . . . .	45
3.3	Modeling asynchronicity using a synchronous Model . . . . .	46
3.3.1	Combined synchronous-asynchronous algorithm . . . . .	46
3.3.2	Computational results . . . . .	47
3.4	Modeling gene perturbations . . . . .	48
3.4.1	Problem formulation . . . . .	49
3.4.2	An algorithm for gene perturbations . . . . .	52
3.5	GenYsis toolbox . . . . .	54
3.6	Case Study: T Helper network . . . . .	54
3.6.1	Simulation results . . . . .	55
3.7	Summary . . . . .	57
<b>4</b>	<b>Multiple Valued Modeling</b>	<b>59</b>
4.1	1-hot encoding of GRNs . . . . .	59
4.2	Multiple valued T Helper network . . . . .	61
4.3	Arabidopsis thaliana network . . . . .	64
4.4	Sigmoid function . . . . .	67
4.5	Combined Boolean-ODE formalism . . . . .	69
4.6	Summary . . . . .	70
<b>5</b>	<b>Probabilistic Boolean Networks</b>	<b>73</b>
5.1	Problem formulation . . . . .	74
5.2	Modified formulation . . . . .	77
5.2.1	Implicit formulation . . . . .	78
5.3	PBN Functionalities . . . . .	79
5.3.1	Algorithm for computing steady state distribution . . . . .	80
5.3.2	Probability of a path . . . . .	81

---

5.3.3	Algorithm for sensitivity analysis . . . . .	83
5.4	Results . . . . .	84
5.5	Summary . . . . .	87
<b>6</b>	<b>Stochasticity in Gene Regulatory Networks</b>	<b>89</b>
6.1	Impact of Stochasticity . . . . .	91
6.1.1	Cellular differentiation . . . . .	91
6.1.2	Robustness of attractors . . . . .	92
6.2	Methods and Techniques . . . . .	93
6.2.1	Fault model . . . . .	94
6.2.2	Stochasticity in nodes . . . . .	95
6.2.3	Stochasticity in functions . . . . .	95
6.2.4	Algorithms for stochastic cellular differentiation . . . . .	98
6.2.5	Algorithm for robustness computation . . . . .	101
6.3	Simulation results . . . . .	101
6.3.1	Cellular differentiation . . . . .	102
6.3.2	Robustness of attractors . . . . .	103
6.4	Summary . . . . .	106
<b>7</b>	<b>Modeling Cell Growth vs. Apoptosis</b>	<b>107</b>
7.1	Cell survival vs. apoptosis GRN . . . . .	108
7.1.1	P53 module . . . . .	109
7.1.2	AKT module . . . . .	111
7.1.3	PI3K module . . . . .	114
7.1.4	mTOR module . . . . .	116
7.2	Modeling growth vs. apoptosis . . . . .	118
7.2.1	p53 mutation . . . . .	119
7.2.2	PTEN mutation . . . . .	120
7.2.3	TSC2 mutation . . . . .	120
7.3	Summary . . . . .	122
<b>8</b>	<b>Conclusions</b>	<b>123</b>
8.1	Thesis summary and contributions . . . . .	123
8.2	Future work . . . . .	126
	<b>Bibliography</b>	<b>129</b>
	<b>Curriculum Vitae</b>	<b>143</b>

# List of Figures

---

2.1	Cell Structure . . . . .	8
2.2	Synthetic Gene Regulatory Network . . . . .	10
2.3	Modeling techniques . . . . .	11
2.4	ODE simulation . . . . .	13
2.5	State Transition Diagram . . . . .	14
2.6	ODE simulation . . . . .	17
2.7	State Transition Diagram . . . . .	18
2.8	Finite State Machine . . . . .	20
2.9	A sample BDD . . . . .	21
2.10	A sample ROBDD . . . . .	21
2.11	ROBDD construction . . . . .	22
2.12	ROBDD representation of a FSM . . . . .	23
2.13	A sample ADD . . . . .	25
2.14	A synthetic GRN . . . . .	26
2.15	Petri net mapping of a GRN . . . . .	29
3.1	Boolean mapping of biological functions . . . . .	32
3.2	Boolean mapping of a GRN . . . . .	33
3.3	Mapped GRN with registers . . . . .	34
3.4	Synchronous state transition Diagram . . . . .	37
3.5	Asynchronous state transition Diagram . . . . .	39
3.6	Synchronous attractors . . . . .	41
3.7	Asynchronous attractors . . . . .	41
3.8	Perturbed GRN . . . . .	49
3.9	Perturbed Boolean function . . . . .	51
3.10	Perturbed GRN . . . . .	52
3.11	T-Helper GRN . . . . .	55
3.12	Results of Gene Perturbation Experiments . . . . .	56
4.1	Multi-valued GRN . . . . .	60
4.2	Multiple valued T-Helper GRN . . . . .	62
4.3	Arabidopsis thaliana GRN . . . . .	64
4.4	Sigmoid function . . . . .	68

---

4.5	Results of Gene Perturbation Experiments . . . . .	70
5.1	A sample PBN . . . . .	74
5.2	A PBN mapping of GRN . . . . .	75
5.3	State Transition Diagram of a PBN . . . . .	76
6.1	Biological functions classification . . . . .	90
6.2	SIN vs. SIF example . . . . .	90
6.3	T-Helper GRN . . . . .	92
6.4	Stochastic cellular differentiation . . . . .	92
6.5	Boolean mapping of a GRN . . . . .	94
6.6	SIN vs. SIF cellular differentiation . . . . .	102
6.7	Robustness of T-Helper GRN . . . . .	104
6.8	Robustness of T-Cell Receptor GRN . . . . .	105
7.1	Apoptosis vs. Cell growth GRN . . . . .	108
7.2	Key proteins in cancer pathways . . . . .	109
7.3	p53 module . . . . .	110
7.4	Simulation results for p53 module . . . . .	111
7.5	AKT pathway . . . . .	112
7.6	Simulation results for AKT module . . . . .	113
7.7	PI3K pathway . . . . .	114
7.8	Simulation results for PI3K module . . . . .	116
7.9	mTor pathway . . . . .	117
7.10	Simulation results for mTOR module . . . . .	117
7.11	Simulating Growth vs. Apoptosis . . . . .	118
7.12	Simulating p53 mutation . . . . .	119
7.13	Simulating PTEN mutation . . . . .	120
7.14	Simulating TSC mutation . . . . .	121



---

# Introduction

---

# 1

## 1.1 Motivation

Living organisms exhibit a complex hierarchical organisation of small building blocks. All self-replicating organisms are composed of cells. All information required for the functioning of a cell is encoded in the *DNA* sequence that is passed on from one cell to another in inheritance. Small fragments of DNA sequence encode the *genes* of an organism. Expression of the genes leads to formation of *proteins*. A combination of these proteins defines the specific functionality of the cell. A group of specialized cells come together to form *tissues*. Different types of tissues form an *organ*. Several organs coordinate with each to function as an *organ system* (such as respiratory system, digestive system, etc). And finally, all the organ systems work synergistically to enable proper functioning of a living organism.

Malfunctioning at any stage of these building blocks may cause a living organism to stop performing normally leading to a diseased state. Most causes of diseases can be mapped to the abnormal activity of some genes in the cells. Normally, cells are equipped to defend themselves against abnormal activity of genes by either killing the defective cells or activating other genes that can neutralise the effectivity of abnormal genes. However, external interference in terms of drug compounds becomes a necessity when cells lose their ability to control abnormal activity of genes leading to uncontrolled cellular malfunctioning which can quickly spread as cells undergo proliferation.

The Human Genome project [83], one of the primary aims of which was to identify all protein coding genes, has estimated and identified approximately 20,000-25,000 protein coding genes in humans. With the identification of all protein coding genes, various high-throughput technologies (such as DNA microarrays [69, 107], protein arrays [47]) have emerged since the completion

of the project. These technologies can measure the expression (or activity) of all genes in a genome simultaneously. By measuring and comparing the expression of genes in an unhealthy vs. a healthy cell, it is now possible to identify genes responsible for various diseases at the entire genome level. High-throughput technologies have been widely adopted by biologists to improve their understanding of living systems and has led to the generation of enormous amounts of experimental data. With an unprecedented amount of biological data coming out of research labs, research focus has shifted from the generation of data to the interpretation and presentation of data in the most efficient manner [97, 54, 55]. Many sophisticated statistical and computational tools have been developed to help biologists identify novel targets from their experimental data.

Predicting a set of responsible genes is often not sufficient as some of the identified genes may be critical for the functioning of cells or it may not be possible to manipulate these genes using drug molecules due to technological reasons. This necessitates studying the behavior of genes with respect to other genes (or proteins) that are known to play a role in a specific disease and can potentially serve as drug targets. This has shifted the focus of computational and experimental tools from just measuring the expression of genes to construction of gene-gene, gene-protein and protein-protein interaction networks [4, 54, 55]. High-throughput technologies such as yeast two hybrid screening arrays [138, 93, 85], DNA microarrays and protein arrays are now increasingly being employed to identify potential interactions among genes and proteins. At the same time, new computational methods are being developed to model and analyse these interaction networks [12].

Interaction networks can represent the dynamic behaviour in terms of the flow of signaling from a biological entity to another and are referred by various names such as *signaling pathways*, *gene regulatory networks* (GRNs) or *genetic regulatory networks*. These GRNs can be seen as dynamical systems of sets of genes and proteins where the gene (or protein) expression is a function of the expression of other genes (and proteins) with which it can directly interact [143]. By modeling the dynamical system represented by GRNs, it is possible to make an *in silico* simulation of the evolution of a gene (and protein) expression over time. Also, one can study how the system behaves when it is slightly changed to reflect gene mutations inside a cell.

The starting point of all modeling efforts is prior knowledge of the system. The complexity of a model depends upon the extent to which the prior knowledge is available (or can be acquired) from experiments. For modeling the behavior of a cell, while a wealth of information is present on the interactions of genes and proteins, the exact stoichiometry and precise kinetics (at the molecular interaction level) still evades our technologies and understanding. In such a situation, one could either wait to gather the crucial information on the precise biochemical processes or choose to model the flow of information in gene interaction networks. In this dissertation, we choose the latter approach as we think that the information available is sufficient to identify



qualitative behavior of the biological system under study. Enabling such kinds of approaches should further our understanding of key elements that dictate the cell fate.

With an improved understanding of gene regulation processes, modeling efforts are increasingly being used for generating the hypotheses that are then tested with experiments. This is gradually changing the outlook towards modeling that was traditionally seen as a platform to communicate experimental outcomes. The same can be observed with respect to modeling GRNs. Qualitative modeling of GRNs has been a research interest to theoretical biologists for at least last three decades [143, 131, 38, 148]. But most of the initial focus of qualitative modeling was on studying theoretical properties of the dynamical nature of GRNs rather than developing computational methods for modeling large GRNs. However in the last decade, a need for efficient computational tools for qualitative modeling of GRNs has been felt so as to understand the experimental data in the context of the dynamical behavior of a cell and generate hypotheses with the assistance of computational tools [117, 36, 119, 106]. A detailed literature survey on qualitative modeling of GRNs is being deferred to Chapter 2 where we first introduce some background principles helpful in understanding different modeling methodologies.

## 1.2 Contributions

The main focus of this research has been on providing computational tools to biologists that can be used for modeling novel hypotheses, such as drug response and gene mutations, purely based on prior knowledge available in the literature about the specific biological system. The models proposed are able to capture complex processes, such as cellular differentiation, growth and apoptosis, based on qualitative knowledge such as gene-protein or protein-protein interactions. This dissertation specifically makes the following contributions:

- A common mathematical framework for Boolean, multiple-valued and stochastic simulations of GRNs has been developed showing the application on some well-studied biological systems such as T-helper cell differentiation and flowering of *Arabidopsis thaliana*.
- Algorithms and computational models have been proposed for efficient discrete modeling of GRNs. Algorithms are based on implicit representation and traversal techniques using Reduced Order Binary Decision Diagrams (ROBDDs) and Algebraic Decision Diagrams (ADDs). Implicit representation and traversal methods facilitate modeling of GRNs with over hundreds of nodes.
- A GRN for modeling interactions among the key players in cancer pathways has been developed. The GRN is able to capture the balance between cell growth and cell apoptosis in the presence of various known

gene mutations in cancer cells and identify drug targets for cancer therapies.

- Algorithms proposed in this thesis are implemented under a common modeling toolbox, `genYsis`. The toolbox `genYsis` has been made available to the research community in the public domain. The software requires no knowledge about the underlying computational methods and facilitates easy-to-use interface for biologists.

### 1.3 Assumptions and limitations

The strongest assumption in this work is on the discrete nature of gene (or protein) regulation. Biological entities are known to evolve in a continuous manner. For example, protein concentrations inside a cell cannot increase in abrupt-short pulses but rather changes uniformly with time. On the other hand, the assumption on discrete regulation of gene or protein expression also serves as a focal point of this research as modeling can be done based on pure qualitative knowledge about interaction between genes or proteins (that is widely available from the existing experimental methodologies).

A second assumption has been made on the availability of prior knowledge on gene-gene, gene-protein and protein-protein interactions. Algorithms proposed in this thesis start with a basic configuration of GRNs, that are in practice, constructed either from the literature or inferred from experimental data (by data mining and machine learning approaches). Automated construction of GRNs is the research focus of a dedicated field of data mining and machine learning. However, this assumption should not be seen as a limitation as algorithms proposed in this thesis are mainly aimed at providing ways to simulate GRNs and can be used for the analysis and evaluation of GRNs resulting from data mining and machine learning approaches.

### 1.4 Thesis organisation

This thesis is divided into four parts. In the first part of the thesis, we outline the background knowledge that is required for understanding the rest of the thesis. The second part of the thesis deals with deterministic Boolean and multi-valued modeling of GRNs. The third part of the thesis extends the algorithms developed for Boolean models to stochastic modeling of GRNs. Finally, in the last part of the thesis, we show the applications of the algorithms developed in this thesis by performing a case study on a GRN that models the delicate balance between cellular growth and apoptosis in a healthy cell vs. a cancer cell.

**Chapter 2** gives a background on GRNs, modeling methodologies in systems biology, relevant past work on Boolean modeling and an introduction to Binary Decision Diagrams.

**Chapter 3** introduces Boolean formulation of GRNs in detail and presents algorithms for modeling the dynamics of GRNs. Mathematical formulation proposed in this chapter is used as a basis for the remaining chapters.

**Chapter 4** extends the Boolean formalism for modeling multiple-valued expression in GRNs. A sigmoid-function based method for generating multiple valued transition rules has been introduced and a methodology for the combined discrete and continuous simulations of GRNs has been discussed.

**Chapter 5** highlights the non-deterministic nature of GRNs due to the presence of multiple alternative biological explanations for the same physiological behavior. Boolean formalism and algorithms proposed in Chapter 3 are extended to *probabilistic Boolean networks* (PBNs) that have been proposed in the past for non-deterministic modeling of GRNs.

**Chapter 6** introduces a new methodology for modeling stochasticity due to malfunctioning of biological phenomena underlying GRNs. Algorithms from Chapter 3 are extended to stochastic Boolean functions and it has been shown that the new stochastic model can give more biologically practical results than a more widely used stochastic modeling technique.

**Chapter 7** presents a case study on the GRN that models a balance between growth and apoptosis signals in a cell, highlighting a practical application of algorithms proposed in the thesis. GRN mapping the key proteins involved in unregulated growth of cells leading to their cancerous behaviour has been constructed in this chapter.

**Chapter 8** concludes the dissertation by giving a more detailed contribution of this research and highlights some possible extensions of this work in other areas of biology such as synthetic biology.



---

# Background

---

# 2

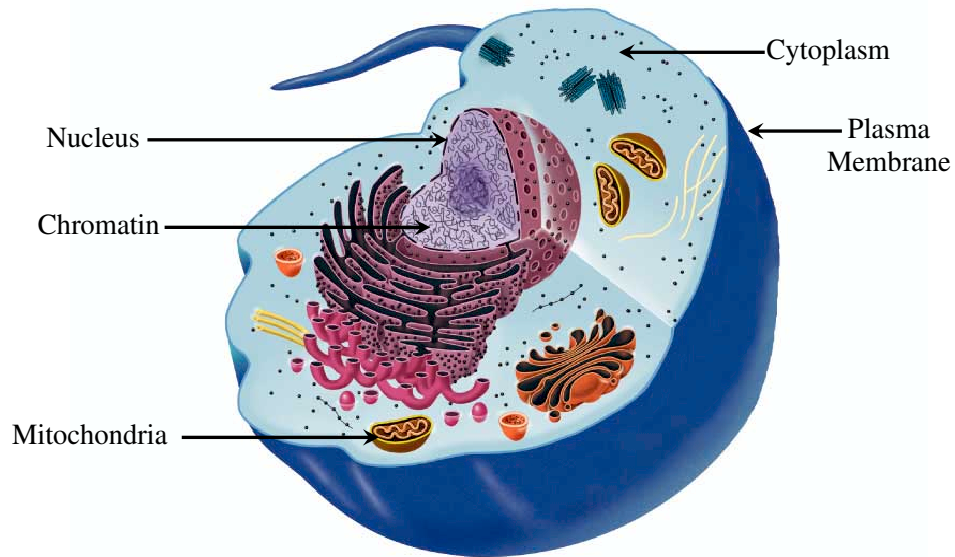
In this chapter we give a background material helpful for understanding the algorithms and methods proposed subsequently in this thesis. We start with an introduction to cellular organisation, intra-cellular signaling components and their corresponding representation using GRNs. Then Sections 2.2 and 2.3 introduces commonly used modeling methodologies for GRNs. Section 2.4, gives a small introduction to Boolean algebra and ROBDDs, which forms a core of most of the algorithms proposed in the thesis. Finally Section 2.5, gives an overview of relevant work on discrete modeling of GRNs.

## 2.1 Biological principles

### 2.1.1 Cellular organisation

Cells consist of various organelles (such as mitochondria, chloroplast and nucleus). Based on the intra-cellular organisation of these organelles, living organisms can be divided into two categories, namely : *prokaryotes* and *eukaryotes*. Prokaryotes are single-celled organisms, and their cell lacks a nucleus. Eukaryotes can be both single-celled or multi-celled organisms; their cells are organised into a more complex configuration and contains a nucleus. Figure 2.1 displays the organisation of a eukaryote cell. A short explanation on functionalities of the different subunits is given below. A more detailed explanation can be found in [16, 118]

**Plasma membrane:** Plasma membrane (or cell membrane) is a semi-permeable bilayer that acts as an interface between the sub-cellular components and the outside environment. Proteins embedded into the membrane act as a communication channel by either allowing few specific molecules to pass



**Figure 2.1:** Cellular organisation [118].

through the membrane or by acting as a binding site (both intra-cellular and extra-cellular) for some other molecules.

**Nucleus:** Nucleus is the regulatory center of the cell. It contains all the genetic information organised as chromatin structures. It has a nuclear membrane with protein embedded pores that acts as a gateway for different molecules to enter the nucleus and regulate chromatin structures.

**Chromatin:** Chromatin is the condensed structure formed by the combination of DNA, RNA and proteins. *DNA* encodes the genetic information as a sequence of nucleotide base pairs: adenine (A), guanine (G), cytosine (C), and thymine (T). Small fragments of the DNA sequence, that encodes a protein are called *genes*.

**Cytoplasm:** Cytoplasm is the liquid material composed of salts, nutrients, enzymes (proteins) etc. It broadly defines the space between the nucleus and the plasma membrane in a cell.

**Mitochondria:** Mitochondria are the bilayer organelles floating inside the cytoplasm. Their main task is to serve the energy requirements of the cell by releasing ATP (*adenosine triphosphate*) which is the cellular energy storage molecule. Mitochondria also have their own DNA (termed as mDNA) and can also participate in various other sub-cellular processes including cell-growth and cell death.

## 2.1.2 Cellular signaling

A cell is the smallest self-sustaining functional unit of living organisms being able to respond to the environment by sensing external signals at its surface and propagating these signals within the surface by functionally activating various sub-cellular proteins. Even though cells can sense a wide variety of environmental signals, mechanisms by which a protein propagates these signals form a very small set and can be categorised into one of the following five types: receptors, kinases, phosphatases, scaffolds and transcription factors [95].

**Receptors:** A receptor is a protein molecule which acts as a binding site for the freely moving signaling molecules either inside or outside the cell membrane. The binding molecule is called a ligand. Receptors propagate signals by undergoing conformational changes after binding to a ligand.

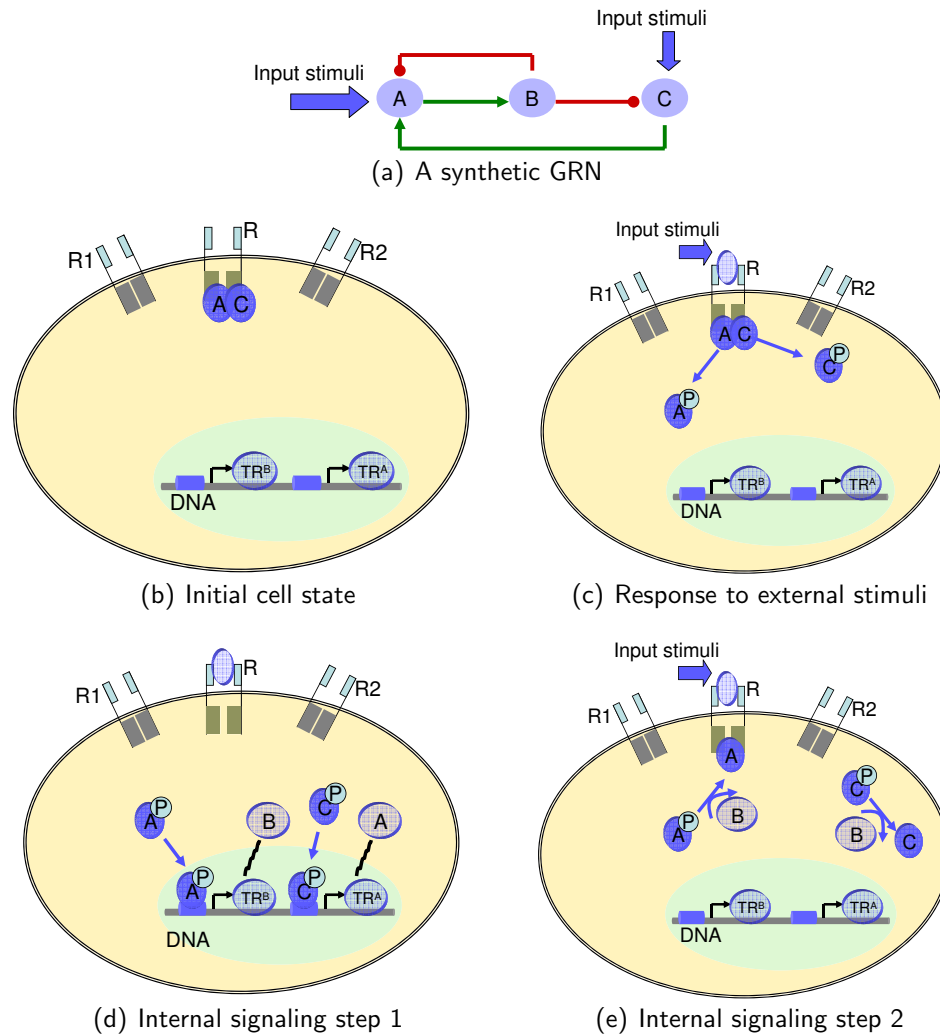
**Kinases:** Kinase proteins can transmit signals by adding a phosphate ( $\text{PO}_4$ ) group to the destination protein. The destination protein is said to become *phosphorylated* on coming in contact with an active kinase. Phosphorylation causes change in the shape of the target protein leading to either activation or deactivation of its functionality.

**Phosphatases:** Phosphatases have the reverse functionality of a kinase. They act by removing the phosphate group from the target protein. Similar to kinases, phosphatases can both activate or inhibit the target protein functionality. The target protein is said to become *de-phosphorylated* on interacting with phosphatase proteins.

**Scaffolds:** Scaffold proteins acts as a platform for bringing together (and assembling) two or more proteins. The assembled complex works by either activating one of the partner proteins or by regulating the activity of a downstream protein.

**Transcription factors:** Transcription factors assist in DNA transcription. They normally reside outside the nucleus and trans-locate to the nucleus on getting activated by another protein. Once inside the nucleus they bind to the promoter region of the protein coding region of the DNA and start the transcription process. Transcribed DNA (called mRNA) trans-locates outside the nucleus and undergoes translation leading to production of protein.

From the above classification, it is evident that even if a protein is always present inside a cell, it may not be in its functional (or activated) form and can exert its biological functionality only when it gets activated by one or more mechanisms listed above. The next section gives an abstraction of these signaling mechanisms as a GRN.



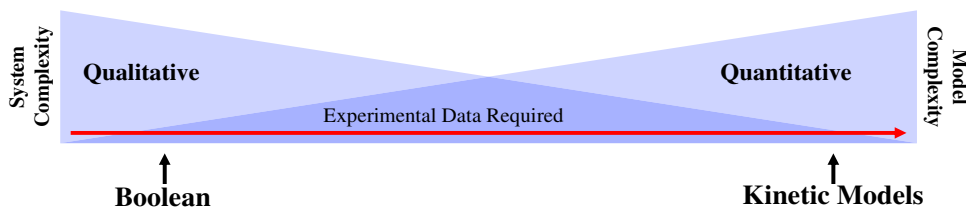
**Figure 2.2:** (a) A Gene Regulatory Network (GRN). (b) A resting cell state. (c)-(e) Cellular signaling in response to input stimuli.

### 2.1.3 Gene regulatory networks

A protein that activates (or inhibits) the functionality of another protein is said to interact with the target protein by an activating (or an inhibiting) mechanism. A graphical representation where the nodes of the graph represents the functional form of proteins and directed edges represent the activation (or inhibition) mechanism is commonly referred to as a *gene regulatory network* (GRN). A small synthetic GRN is represented in Figure 2.2(a), where activating edges are represented by arrow-headed lines and inhibiting edges are represented by circle-headed lines.

Biological phenomena underlying different interactions in the GRN of Figure 2.2(a) are shown in Figures 2.2(b)-2.2(e). Figure 2.2(b) represents the cell state in the absence of any input stimuli. When an input stimuli is sensed at





**Figure 2.3:** Different modeling techniques for simulating GRNs categorised on the basis of their complexity and the complexity of the systems they can be modeled [adapted from [125]].

a surface receptor (Figure 2.2(c)), proteins  $A$  and  $C$  get dissociated from the membrane and get phosphorylated. In the next time instance (Figure 2.2(d)), phosphorylated  $A$  and  $C$  can transcriptionally express protein  $B$  and  $A$  respectively. This is represented in the GRN by activating edges from  $A$ -to- $B$  and  $C$ -to- $A$ . Finally (in Figure 2.2(d)), expression of protein  $B$  de-phosphorylates  $A$  and  $C$  leading to loss of their functionality. This mechanism is represented by inhibiting edges from  $B$ -to- $A$  and  $B$ -to- $C$ .

The GRNs, such as the one in Figure 2.2(a), summarises known interactions among a set of proteins that participate in a given biological phenomenon. Such a GRN represents a dynamical system where a node changes its value over time depending upon the state of the neighbouring nodes in the network. By modeling these GRNs as a dynamical system one can capture some aspects of cellular dynamics. The next section summarises some of the modeling techniques that are widely used for studying GRNs in Systems biology.

## 2.2 Modeling techniques

The GRN modeling techniques can be broadly categorised into continuous and discrete approaches. Figure 2.3 summarizes advantages and disadvantages of these two modeling approaches. In this section we first explain, with the help of a small example, the two modeling approaches. Then in the end, we compare and contrast advantages and disadvantages of both the modeling approaches.

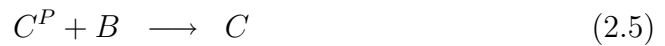
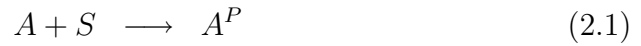
### 2.2.1 Continuous modeling

Continuous models (or Kinetic models) such as *ordinary differential equations* (ODEs) and *partial differential equations* (PDEs) are used for modeling the continuous evolution of the concentration of proteins in a GRN. These models use parameters such as rate constant of chemical reactions, membrane diffusion constants and protein degradation rates to simulate the kinetics of the system. An ODE formulation of the GRN in Figure 2.2(a) is explained in Example 2.2.1.

Rate Constants		Initial Conditions	
$k_{s,a}$	0.6	$S_0$	5
$k_{s,c}$	0.6	$A_0$	5
$k_{a,b}$	0.8	$A_0^P$	0
$k_{b,a}$	0.5	$B_0$	0
$k_{b,c}$	0.5	$C_0$	5
$k_{c,a}$	0.2	$C_0^P$	0

**Table 2.1:** Parameters used in ODE.

**Example 2.2.1** Interactions among the nodes in the GRN of Figure 2.2(a) can be specified in further details by using the following set of chemical reactions, where  $A^P$  and  $C^P$  stands for the phosphorylated form of  $A$  and  $C$  respectively:

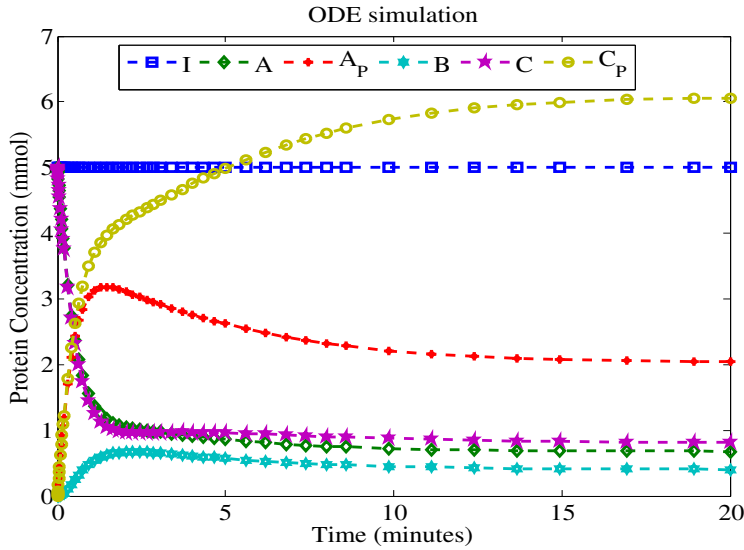


If the concentration of proteins  $A$ ,  $A^P$ ,  $B$ ,  $C$ ,  $C^P$  and the input stimuli  $S$  are given by  $x_a$ ,  $x_{a^p}$ ,  $x_B$ ,  $x_C$ ,  $x_{c^p}$  and  $x_S$  respectively, then the rate of evolution of these proteins can be specified by using the set of coupled ODEs in Equation 2.7. Parameter  $k$ 's in Equation 2.7 represents the rate constants of different interactions in the GRN and are specified as in Table 2.1.

$$\begin{bmatrix} \frac{dx_s}{dt} \\ \frac{dx_a}{dt} \\ \frac{dx_{a^p}}{dt} \\ \frac{dx_b}{dt} \\ \frac{dx_c}{dt} \\ \frac{dx_{c^p}}{dt} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ -k_{s,a} & 0 & 0 & k_{c,a} & 2k_{b,a} & 0 \\ k_{s,a} & 0 & -k_{a,b} & 0 & -k_{b,a} & 0 \\ 0 & 0 & k_{a,b} & 0 & -k_{b,a} & -k_{b,c} \\ 0 & -k_{s,c} & 0 & 0 & 0 & 2k_{b,c} \\ 0 & k_{s,c} & 0 & -k_{c,a} & 0 & -k_{b,c} \end{bmatrix} \cdot \begin{bmatrix} x_s \cdot x_a \\ x_s \cdot x_c \\ x_{a^p} \\ x_{c^p} \\ x_{a^p} \cdot x_b \\ x_{c^p} \cdot x_b \end{bmatrix} \quad (2.7)$$

Given the initial concentrations of these proteins as in Table 2.1, Figure 2.4 gives the simulation results for the ODE's of the small synthetic network. As one can see from the simulation results, protein concentration stabilises over time. The stabilised concentration of proteins reflect the steady state of the GRN. ■

Kinetic models have been applied successfully in the past for modeling various GRNs [17, 91, 141]. However, their applications have been restricted



**Figure 2.4:** Dynamic simulation of the ODE formulation of the GRN in Figure 2.2(a).

to only small well studied GRNs due to the fact that kinetic parameters are not known for most of the interactions in a GRN. Kinetic parameters for all the gene-gene or gene-protein interactions in a GRN are rarely available from the literature and experimentally measuring these parameters is currently infeasible using the available experimental techniques.

## 2.2.2 Discrete modeling

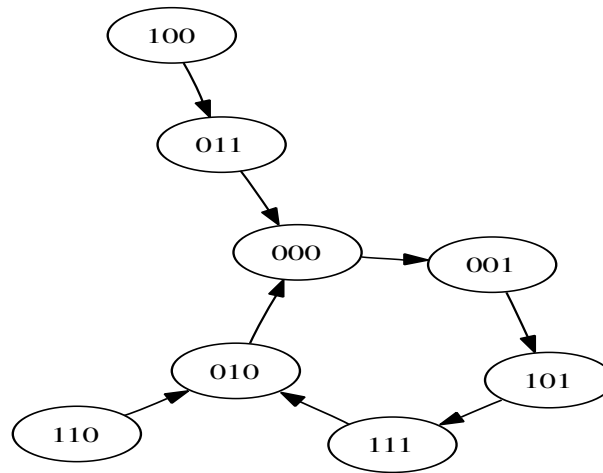
In the discrete modeling of GRNs, a node can take only discrete expression values, unlike the continuous range of values that are possible in kinetic models. In the most restrictive case, a node exists in only two expression states to represent *active* and *inactive* gene (or protein) represented by Boolean 1 and 0 respectively. In Boolean models, an inhibiting edge can change the expression of the node from 1-to-0 and an activating edge can change the state from 0-to-1 (only when no inhibition is present). A snapshot of the activity level of all the nodes in the GRN at a given time instance is called the state of the network.

**Example 2.2.2** For the GRN in Figure 2.2(a), the network can exist in one of the states shown in Figure 2.5 and can make a transition from one state to another state based on the Boolean functions defined in Equations 2.8-2.10, where the symbols  $\wedge$  and  $\neg$  stands for Boolean AND and NOT respectively.

$$x_a = x_c \wedge \neg x_b \quad (2.8)$$

$$x_b = x_a \quad (2.9)$$

$$x_c = \neg x_b \quad (2.10)$$



**Figure 2.5:** State Transition Diagram corresponding to Boolean model of GRN in Figure 2.2(a). Every vertex represents a state of the network as defined by the expression of proteins  $A$ ,  $B$  and  $C$ .

A cycle formed by the set of network states  $\{000, 001, 101, 111, 010\}$  is called a *dynamical attractor*. There can be  $2^N$  network states if there are  $N$  nodes in the GRN. Exponential state space of Boolean models of a GRN, makes this modeling technique both computationally and memory intensive. ■

In Boolean modeling, kinetic parameters are not required to define interactions between the genes (or proteins). However, such a simplification comes at the cost of discretization of the gene expression (or protein concentrations) to only two expression levels, namely: *present* or *absent*. Nevertheless Boolean modeling can efficiently capture the required dynamics of a GRN and has been successfully applied in the past to model various biological phenomena such as cellular differentiation and embryo development [114, 37, 75, 41].

Boolean modeling of GRNs can be extended to more than two levels of discretization, e.g. low, medium and high. With multiple valued modeling, it is possible to model a scenario where multiple interactions acting on a single gene (or protein) may have relatively different effectiveness in activating the target. However, the computational complexity of the model increases as the number of discretization levels are increased and also more experimental information is required to describe the rules for transitions among different activation levels.

### 2.2.3 Model complexity vs. System complexity

As we have seen from the description of continuous and discrete models above, one needs a significantly more detailed information about the gene-gene, gene-protein or protein-protein interactions in the continuous models as compared to discrete models. This makes the continuous models such as ODEs more complex to build than the discrete models such as Boolean networks. Model

complexity along with the availability of experimental data determines the complexity of biological phenomena (or the system complexity) that can be simulated using that model. While the largest systems that have been successfully modeled using ODEs had at most twenty nodes in a GRN, Boolean modeling has been applied on GRNs with over hundreds of nodes.

### 2.2.4 Quantitative vs. Qualitative

Applications of continuous and discrete models also depends upon the desired output from the modeling effort. Kinetic models are more suitable for modeling the quantitative aspects such as to measure the concentration of drug required for a desired cellular response or to measure the amount of protein produced. Whereas, Boolean models are useful in studying the qualitative aspects of a biological phenomena such as what are the active proteins in a steady state or how the steady state proteins activity changes on gene knock-down and over-expression.

In practice, no single modeling technique can capture all aspects of a biological phenomenon and multiple methods are often used during different phases of hypothesis generation and testing.

## 2.3 Non-determinism in GRNs

The process of gene activation (or inhibition) is a complex mechanism that involves binding of one or more transcription factors, transcription of DNA into mRNA and translation of mRNA into protein. All these stages of gene activation involve some amount of stochasticity. The level of stochasticity induced depends upon the complexity of the binding mechanism, length of the protein coding regions, concentration of transcription factors and various other factors. Experimentally, gene regulation processes have been shown to be inherently stochastic [62, 2, 113, 111, 87, 130]. In the presence of stochasticity, a gene may not get expressed even in the presence of required transcription factors, leading to stochastic fluctuations in the states of the GRN. These stochastic fluctuations may cause the dynamics of identical cells to behave differently in the same environmental conditions.

### 2.3.1 Modeling stochasticity in continuous models

Traditionally, the stochasticity in GRNs is modeled as associated to the concentration of the reacting species. At low concentrations of reacting species, the probability of two molecules undergoing a biochemical reaction decreases, thereby adding a stochastic effect on the reaction product concentration. This approach of noisy gene regulation can be efficiently simulated in continuous modeling approaches by using *chemical master equations* (CMEs) and Gillespie's algorithm [33, 34, 22, 24, 26].

**Algorithm 1:** Gillespie's algorithm.

---

```

1 GILLESPIE( $T_{max}, \mathbf{x}_0, \Delta, h_r$ )
2 begin
3    $\mathbf{x} \leftarrow \mathbf{x}_0$ 
4    $t = 0$ 
5   while true do
6     draw  $T \sim \text{Expo}(\sum_r h_r(\mathbf{x}))$ 
7      $t = t + T$ 
8     if  $t > T_{max}$  then
9       break
10    draw  $R$  in set of reactions according to probability distribution
11     $\mathbb{P}\{R = r_0\} = \frac{h_{r_0}(\mathbf{x})}{\sum_r h_r(\mathbf{x})}$ 
12     $\mathbf{x} \leftarrow \mathbf{x} + \Delta_R$ 
13  return  $\mathbf{x}$ 
14 end

```

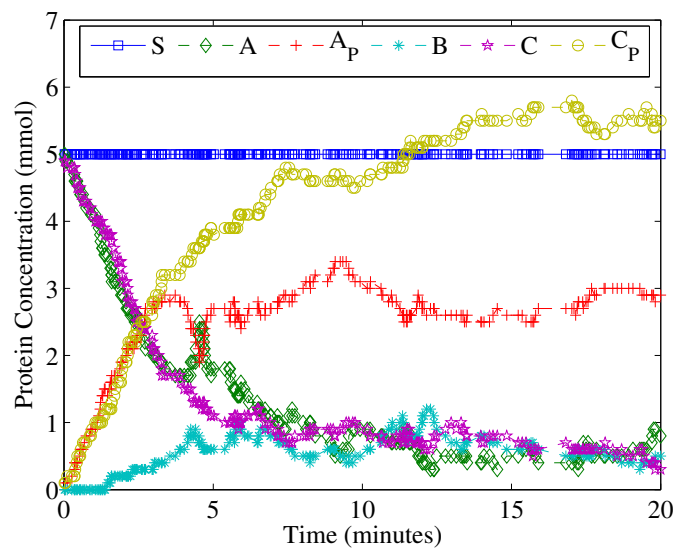
---

Gillespie's algorithm in its most basic form is described by Algorithm 1. In this algorithm,  $h_r$  defines the rate of each reaction  $r$ ,  $T_{max}$  is the amount of time for which the simulation is required,  $\mathbf{x}_0$  is the initial concentration of proteins (or reactants) in terms of number of molecules and  $\Delta$  represents the amount by which the reactant concentration changes at a specific time step. In Line 5, the next time instance when some reaction takes place is selected. This time step is drawn from an exponential distribution defined by the sum of rates of all the reactions. Then, the reaction which will occur at this time instance is selected in Line 11. The reactant concentrations are updated according to given  $\Delta$  and the steps are iterated until the simulation time exceeds  $T_{max}$ . For a detailed reading on Gillespie's algorithm and CME's, one is referred to [33, 34, 79].

**Example 2.3.1** The discrete stochastic simulation using the Gillespie's algorithm for the set of reactions introduced in Example 2.2.1, is as shown in Figure 2.6. For this simulation, the same rate constants and initial concentrations were used as in Example 2.2.1 and  $\Delta$  was taken as 0.1. The rate of reactions  $h_r$  are given by:



As one can see from Figure 2.6, the protein concentration changes discretely



**Figure 2.6:** Dynamic stochastic simulation using Gillespie's algorithm for the GRN in Figure 2.2(a).

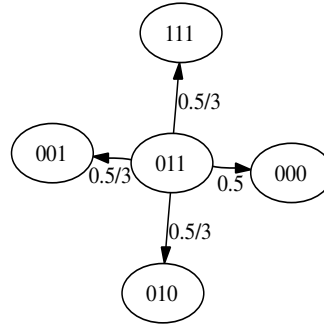
over time and stabilises at almost the similar concentrations as by using the non-stochastic simulations in Figure 2.4. ■

Just like continuous modeling of GRNs, stochastic modeling using Gillespie's algorithms requires that all the kinetic rate constants are known a priori. This requirement restricts its application to only small well studied networks.

### 2.3.2 Modeling stochasticity in discrete models

To simulate the stochastic effects due to low concentrations of reacting species in Boolean models, methods are proposed in the literature where the nodes in the GRN are flipped from 0 to 1 or vice versa with some predefined flip probability [14, 90, 39, 114]. With this model of generating stochasticity in Boolean models, differentiation into multiple steady states can be simulated. This model of stochasticity is referred to by the term *Stochasticity in Nodes* (or SIN) in this dissertation. Unlike the CME approach, the SIN model of stochasticity does not take into consideration the correlation between the expression values of reacting species and the probability of flipping the expression of a node due to noise. Further, the SIN approach models the stochasticity at a node regardless of the susceptibility to stochasticity of the underlying biological function that leads to its activation.

**Example 2.3.2** If every node in the GRN in Figure 2.2(a) can flip with a probability 0.5, then the state transition diagram of Figure 2.5 will have few extra edges showing the possible transitions under stochasticity and a probability will be associated with all the edges. For example, Figure 2.7



**Figure 2.7:** State Transition Diagram corresponding to Boolean model of GRN in Figure 2.2(a).

shows the possible next states of the state 011 in the presence of stochasticity.

■

## 2.4 Boolean algebra and implicit methods

In this section, we give a short introduction to Boolean algebra, finite state machines and their implicit representation and traversal using BDDs. This section aims at giving an overview of the concepts required for the self sufficiency of this dissertation. For a detailed introduction to Boolean algebra, finite state machines and BDDs, a reader is referred to [133, 63, 64, 46, 155]. Here, we adapt the formalism and terminology introduced by De Micheli in [46].

### 2.4.1 Boolean algebra

A *Boolean algebra* is defined by the set  $B = \{0, 1\}$  and by two operands  $\vee$  and  $\wedge$  representing the *disjunction* and *conjunction* (often referred to as *sum* and *product* or OR and AND). Traditionally, Boolean values 0 and 1 are often referred to as *false* and *true* respectively.

The multi-dimensional space spanned by  $n$  binary-valued Boolean variables is denoted by  $B^n$ . Every point in this space is defined by a binary vector of dimension  $n$ . Every dimension of the space is said to be represented by a Boolean variable. A *literal* is defined as an instance of a variable or its complement. Product of literals is called a *cube*. The complement of a Boolean variable  $a$  has one of the following representations:  $\bar{a}$ ,  $\neg a$  or  $a'$ .

A *completely specified* Boolean function is a mapping between Boolean spaces. An  $n$ -input and  $m$ -output function is a mapping  $f : B^n \rightarrow B^m$ . In a completely specified function every input Boolean vector maps to a single output Boolean vector. In an *incompletely specified* Boolean function, an input vector can map to more than one output vector and is represented by  $f : B^n \rightarrow \{0, 1, -\}^m$ , where the symbol “-” denotes a *don't care* condition and



can take values in both 0 and 1. Any function can be either expressed as a *sum of product* of  $n$  literals or as a *product of sum* of  $n$  literals.

**Definition 1** *The cofactor of Boolean function  $f(x_1, x_2, \dots, x_n)$  with respect to variable  $x_i$  is  $f_{x_i} = f(x_1, x_2, \dots, x_{i-1}, 1, \dots, x_n)$  and with respect to variable  $x'_i$  is  $f_{x'_i} = f(x_1, x_2, \dots, x_{i-1}, 0, \dots, x_n)$ .*

The cofactor  $f_{x_i}$  is also referred to as a *positive cofactor* and  $f_{x'_i}$  as a *negative cofactor*.

**Definition 2** *The Boolean difference of a Boolean function  $f(x_1, x_2, \dots, x_n)$  with respect to variable  $x_i$  is defined as  $\frac{\partial f}{\partial x_i} = f_{x_i} \oplus f_{x'_i} = f_{x_i} f'_{x'_i} + f'_{x_i} f_{x'_i}$ .*

The Boolean difference  $f_{x_i}$  indicates whether the function  $f$  is sensitive to changes in input  $x_i$ . When it is zero the function does not depend on the variable  $x_i$ .

**Definition 3** *The smoothing or existential quantification of a Boolean function  $f(x_1, x_2, \dots, x_n)$  with respect to variable  $x_i$  is defined as  $\exists_{x_i} f = f_{x_i} + f_{x'_i}$ .*

The smoothing (or existential quantification) of a function with respect to a variable amounts to deleting all appearances of that variable from the function.

**Definition 4** *The consensus or universal quantification of a Boolean function  $f(x_1, x_2, \dots, x_n)$  with respect to variable  $x_i$  is defined as  $\forall_{x_i} f = f_{x_i} \cdot f_{x'_i}$ .*

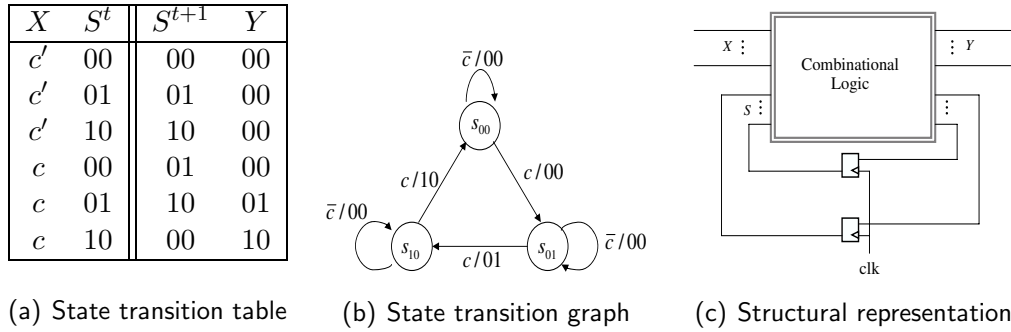
The consensus (or universal quantification) of a function with respect to a variable represents the components of the function that are independent of that variable.

**Definition 5** *The logic equivalence function  $\leftrightarrow$  over two Boolean variables  $a$  and  $b$  is defined as  $a \leftrightarrow b = a \oplus \bar{b} = ab + \bar{a}\bar{b}$ , where  $\bar{a}$  and  $\bar{b}$  represent the complement of variables  $a$  and  $b$ .*

A function  $f = a \leftrightarrow b$  is true if and only if either both  $a$  and  $b$  are true or both  $a$  and  $b$  are false. In other words, the variable  $a$  is said to take the value defined by the variable  $b$ .

## 2.4.2 Finite state machines

A *finite state machine* (FSM) is defined by a 6-tuple  $(X, Y, S, \delta, \lambda, s^0)$ , where  $X$  is the set of input patterns,  $Y$  is the set of output patterns,  $S$  is the set of states,  $\delta : X \times S \rightarrow S$  is the state transition function,  $\lambda : X \times S \rightarrow Y$  is the output function and  $s^0$  is the initial state (or the reset state). Based on the output function  $\lambda$ , FSMs can be classified into *Mealy* or *Moore* machines. In a Moore FSM, output function does not depend upon the input patterns, i.e.



**Figure 2.8:** (a) State transition table representation of a FSM. (b) Corresponding state transition graph. (c) Structural representation of FSM.

$\lambda : S \rightarrow Y$ . In a Mealy FSM, output depends upon both the current state and the input pattern, i.e.  $\lambda : X \times S \rightarrow Y$ .

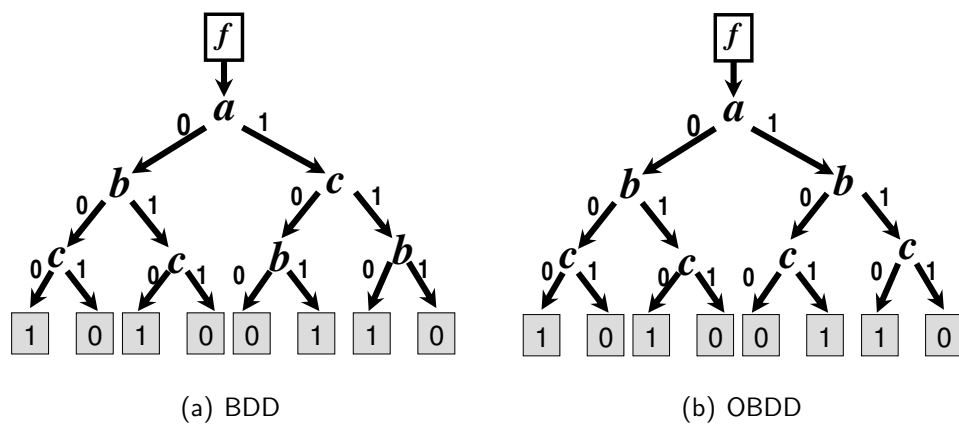
An FSM can have either a *behavioural* or a *structural* representation. A behavioural representation of an FSM can be either using a state transition table (such as the one in Figure 2.8(a)) or as a state transition graph (see Figure 2.8(b)). The structural representation of an FSM is given by a combinational logic and a set of registers (to store the state variables) as in Figure 2.8(c). The behavioral description of an FSM explicitly represents all the states of the machine and their possible transitions. Therefore, explicit representation of FSMs is possible only for small state space. On the other hand, in the structural representation, the state transition function  $\delta$  is implicitly represented by a Boolean function.

### 2.4.3 Binary decision diagrams

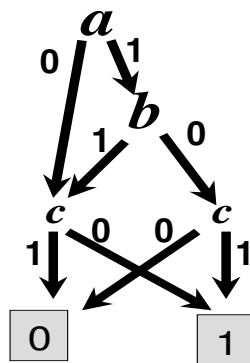
A *binary decision diagram* (BDD) [133] is a rooted directed acyclic graph with two terminal nodes labeled 0 or 1 (representing *false* and *true* values respectively), and a set of intermediate nodes with out-degree two. Each intermediate node is labeled with a binary variable name and the two outgoing edges represent variable evaluation to 0 and 1. The path from the root to the leaf node 1 gives the variable assignment that makes the function evaluates to true. A sample BDD corresponding to the function  $f : c \leftrightarrow (a \wedge \neg b)$  is shown in Figure 2.9(a).

A BDD is ordered (OBDD) if all the paths from the root to the leaf nodes have a same linear order of variables  $x_1 < x_2 < \dots < x_n$ . An OBDD corresponding to BDD in Figure 2.9(a) with the variable order  $a < b < c$  is shown in Figure 2.9(b).

An OBDD is a *reduced ordered BDD* (ROBDD) if no two nodes in the graph have the same sub-graph and no node has a 1-successor similar to its 0-successor. ROBDD corresponding to OBDD in Figure 2.9(b) with the variable order  $a < b < c$  is shown in Figure 2.10. ROBDDs gives an advantage over



**Figure 2.9:** (a) BDD corresponding to Boolean function  $f : c \leftrightarrow (a \wedge \neg b)$ . (b) Corresponding OBDD with the variable order  $a < b < c$ .

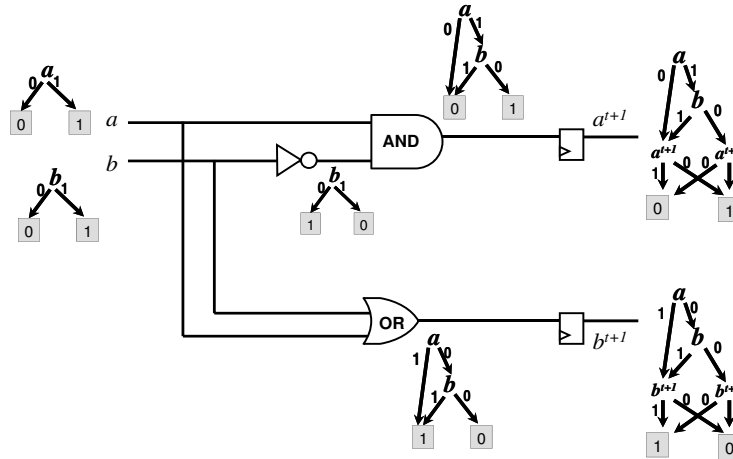


**Figure 2.10:** An ROBDD corresponding to Boolean function  $f : c \leftrightarrow (a \wedge \neg b)$  with the variable order  $a < b < c$ .

BDDs due to their following two characteristics :

- ROBDDs are more efficient than BDDs as redundant nodes are eliminated.
- For a given variable order, an ROBDD is a canonical representation of a Boolean function.

ROBDD representation can be space efficient for most Boolean functions and Boolean operations like AND, OR, existential quantification (i.e.  $\exists$ ) and universal quantification (i.e.  $\forall$ ) can be performed in time that has a polynomial complexity [63] with the number of ROBDD nodes. In a worst case scenario number of nodes in a ROBDD can grow exponentially with the problem size (i.e. number of variables). However, in practice it has been seen that the number of nodes in a ROBDD grows mildly with the problem size for most practical cases. Any Boolean function can be represented using ROBDDs.



**Figure 2.11:** An ROBDD construction from a Boolean function.

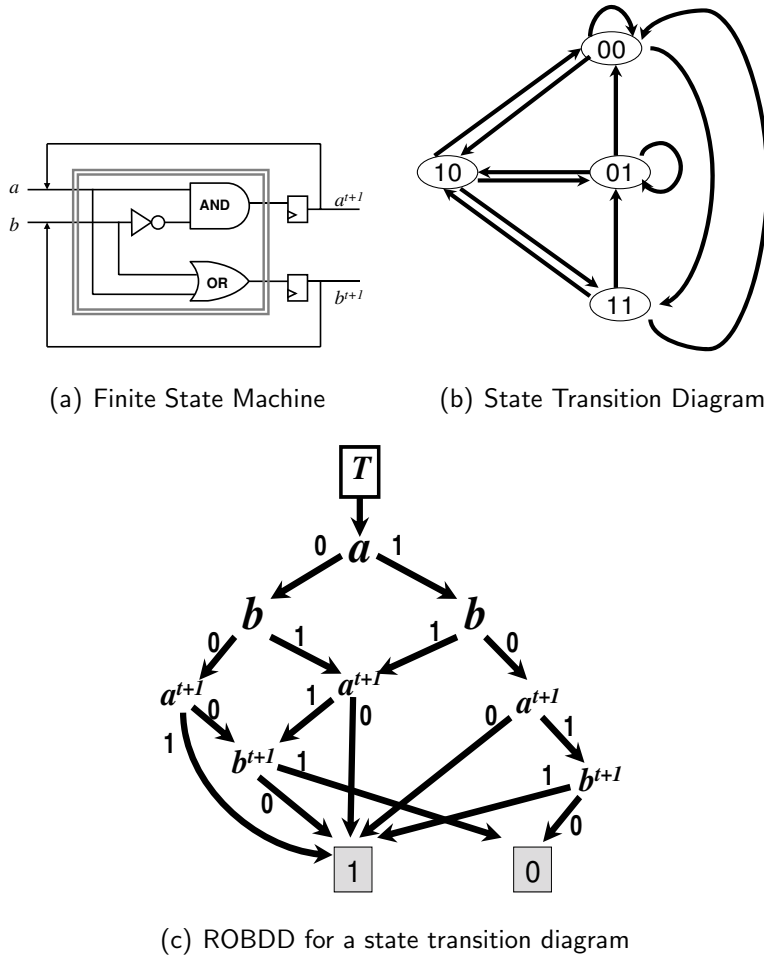
Figure 2.11, demonstrates an example of how a complex Boolean function (represented as a combinational digital circuit) can be mapped into a ROBDD.

In Figure 2.11, if the variable  $a^{t+1}$  is assigned to the input  $a$  at the next time step (and same for  $b$ ), then the input-output activity of the circuit can be seen as an FSM in Figure 2.12(a). Since the state transition function is defined by a Boolean function, the FSM behavior can also be represented using ROBDDs. The corresponding ROBDD for the FSM in Figure 2.12(a) is shown in Figure 2.12(c). The corresponding state transition diagram is shown in Figure 2.12(b). The number of states in a state transition diagram can grow exponentially with the number of state variables. ROBDDs serves as an efficient way to represent all state transitions and can be constructed from the FSMs without explicit enumeration of all states. ROBDD representation of the FSM is also called an implicit representation of the state transition function and is very suitable for reachability analysis on the state space.

#### 2.4.4 Reachability analysis

Given an initial state  $s^0$  of the FSM  $(X, Y, S, \delta, \lambda, s^0)$ , the set of next states under the state transition function  $\delta$  is called the *image* of the state  $s^0$  (represented by  $I_\delta(s_0)$ ). By iteratively computing the image one can list all the possible reachable states from an initial state. The computed reachable states can be subjected to further analysis such as testing for undesirable output state, oscillating behavior etc. Such reachability computation and analysis can be efficiently performed on the ROBDD representation. The image of a state can be determined by following paths from the root node to the 1-terminal node in the corresponding ROBDD.

**Example 2.4.1** Given an initial state 00 (i.e.  $a = 0$  and  $b = 0$ ), following the left-most path in the ROBDD of Figure 2.12(c) one can quickly discover



**Figure 2.12:** (a) A finite state machine. (a) Corresponding state transition diagram. (b) ROBDD representation of the state transition diagram.

that the next states belong to the set of states  $\{10, 11, 00\}$ . The image of the state  $I_0 = 00$  (or the states reachable from  $I_0$  in one step) is given by the set  $RS_1 = \{10, 11, 00\}$ . ■

Many problems in digital system design, combinatorial optimization, artificial intelligence, cryptography can be formulated in terms of operations over Boolean functions. ROBDDs have been effectively applied on a range of problems in these domains. In this thesis, we explore the applications of ROBDDs for implicit modeling and representation of GRNs.

### 2.4.5 Discrete functions and ADDs

Given a set  $V$  of discrete real numbers, an  $n$ -input and  $m$ -output *discrete function*  $f : V^n \rightarrow V^m$  is a mapping from the finite space  $V^n$  to the space defined by  $V^m$ . In this dissertation, we are only interested in discrete functions

with Boolean inputs and outputs defined by a set of finite real numbers, i.e.  $f : B^n \rightarrow V^m$ .

Discrete functions with real numbers as outputs can be represented by an extension of ROBDDs called *algebraic decision diagrams* (ADDs) [135, 44]. Unlike ROBDDs, the leaf nodes in ADDs can take the values in the set defined by  $V$ . For example, ADD for the discrete function  $f = 5 \cdot c \cdot a \cdot \bar{b} + 3 \cdot \bar{c} \cdot \bar{a} + 2 \cdot \bar{c} \cdot b$  is as shown in Figure 2.13(a)

Boolean existential quantification  $\exists_x$  can be generalised to arithmetic existential quantification  $\setminus_x^{op}$  for an arithmetic operation  $op$  such as  $+$ ,  $-$ ,  $\times$ , Max and Min. We will only require arithmetic existential quantification with respect to the sum operator (i.e.  $+$ ) and the Max operator in this dissertation. These operators are defined as in Equations 2.17 and 2.18.

$$\setminus_x^+ f(x, y) = \sum_x f(x, y) \quad (2.17)$$

$$\setminus_x^{Max} f(x) = \max_x f(x) \quad (2.18)$$

In ADD representation,  $\setminus_x^+$  corresponds to removing the BDD nodes corresponding to variable  $x$  and merging its subgraphs. Conversely,  $\setminus_x^{Max}$  corresponds to selecting the leaf node with the largest value.

**Example 2.4.2** Given the discrete function  $f = 5 \cdot c \cdot a \cdot \bar{b} + 3 \cdot \bar{c} \cdot \bar{a} + 2 \cdot \bar{c} \cdot b$ ,  $\setminus_b^+ f(a, b, c)$  is given by :

$$\setminus_b^+ f(a, b, c) = \sum_b f(a, b, c) \quad (2.19)$$

$$= f_{b=0} + f_{b=1} \quad (2.20)$$

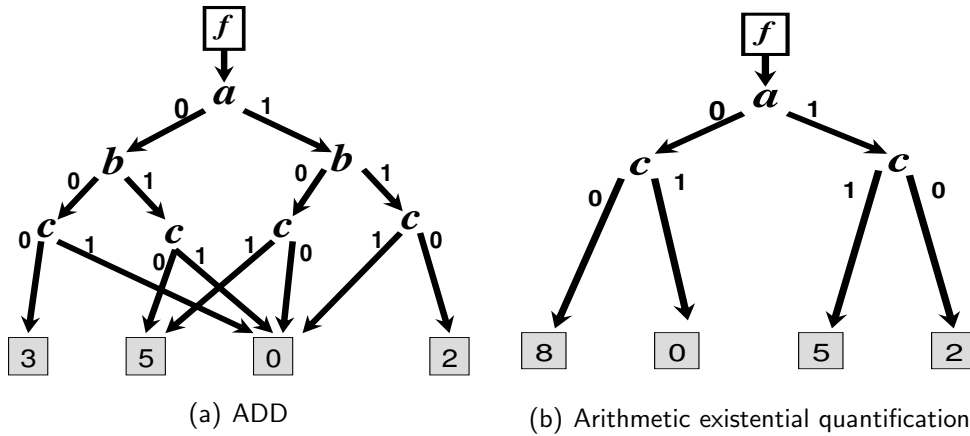
$$= \{5 \cdot c \cdot a + 3 \cdot \bar{c} \cdot \bar{a}\} + \{3 \cdot \bar{c} \cdot \bar{a} + 2 \cdot \bar{c}\} \quad (2.21)$$

$$= 5 \cdot c \cdot a + 8 \cdot \bar{c} \cdot \bar{a} + 2 \cdot \bar{c} \cdot a \quad (2.22)$$

The ADD corresponding to  $\setminus_b^+ f(a, b, c)$  is shown in Figure 2.13(b). The operator  $\setminus_{a,b,c}^{Max} f(a, b, c)$  will return 5 (i.e. the leaf with the largest value in the ADD of Figure 2.13(a)). ■

## 2.5 Previous work

Boolean modeling of GRNs has been addressed quite extensively in the past. However, most of the formalisms of Boolean models either required enumeration of states in some form or they were not suitable for common modeling framework for both deterministic and stochastic modeling of GRNs. The past work on Boolean modeling can be well explained by charting the history of its progress in the last few decades.



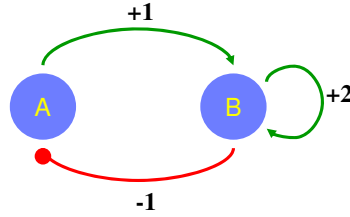
**Figure 2.13:** (a) ADD corresponding to Boolean function  $f = 5 \cdot c \cdot a \cdot \bar{b} + 3 \cdot \bar{c} \cdot \bar{a} + 2 \cdot \bar{c} \cdot b$ . (b) ADD corresponding to  $\int_b^+ f(a, b, c)$ .

### 2.5.1 Explicit enumeration based approaches

Boolean networks for modeling GRNs were first introduced by Kauffman [143] for qualitative analysis of cellular differentiation. Most of the early work on Boolean modeling was targeted at theoretical analysis of GRNs, such as the size and number of attractors with respect to the number of Boolean functions for every node or the number of feedback loops in the GRN. All the initial studies performed explicit enumeration of states in a network so as to identify all steady states.

To avoid enumeration of all states, a concept of loop characteristic states was introduced in [131, 38] by using the principle of *generalized logic analysis (GLA)*. The GLA was essentially a generalisation of Boolean-synchronous transition models of Kauffman in [143] to discrete-asynchronous modeling. A GLA model of GRN can be characterized by the following conditions :

1. States are updated asynchronously (i.e. only one node can change its expression between two consecutive time steps).
2. Each node can have multiple levels of expression (i.e.  $0, 1, 2, \dots$ ).
3. In the state transition diagram, nodes can have threshold values between two consecutive expression states (i.e.  $x_i = \{0, \theta^1, 1, \theta^2, 2, \dots\}$ ). States involving only integer values (i.e.  $0, 1, 2, \dots$ ) are called *regular states* (e.g. 00, 01, etc.), whereas states involving one or more threshold values are called *singular states* (e.g.  $0\theta^1, \theta^2\theta^1$ , etc.).
4. Logical parameters specify the rules of interactions (e.g., if a node  $i$  can have three incoming edges from  $j$ ,  $k$  and  $l$ ; then  $K_{i,i}$ ,  $K_{i,j}$ ,  $K_{i,jk}$  specify relationship between  $x_i$ ,  $x_j$  and  $x_k$ ). These parameters can take the same values as the corresponding variable  $x_i$ .



**Figure 2.14:** A small synthetic Gene Regulatory Network.

**Table 2.2:** State transition table for the GRN in Figure 2.14.

$x_a$	$x_b$	$X_a$	$X_b$
0	0	$K_{1,2}$	$K_{2,}$
0	1	$K_{1,}$	$K_{2,}$
0	2	$K_{1,}$	$K_{2,2}$
1	0	$K_{1,2}$	$K_{2,1}$
1	1	$K_{1,}$	$K_{2,1}$
1	2	$K_{1,}$	$K_{2,12}$

**Example 2.5.1** Let us consider a small GRN in Figure 2.14. Since, the node  $A$  has only one outgoing edge, it is modeled at two logic levels: 0 and 1. The node  $B$  has two outgoing edges and therefore modeled at three expression levels: 0, 1 and 2. The node  $B$  deactivates the node  $A$  at the expression level 1 and activates itself at the expression level 2. The generalised transition rules for the GRN in Figure 2.14 are then specified using the logic Equations 2.23-2.24, where  $d_x$  stands for discretisation of the real values inside the brackets. The capital letters  $X_a$  and  $X_b$  stands for the value of  $A$  and  $B$  at the next step and  $k$ 's are the real-valued parameters.

$$X_a = d_x(k_{21} \cdot x_b) \quad (2.23)$$

$$X_b = d_x(k_{12} \cdot x_a + k_{22} \cdot x_b) \quad (2.24)$$

In the above equations,  $X_a$  can take the values in the set  $\{0, d_x(k_{21})\}$  and  $X_b$  can take the values in the set  $\{0, d_x(k_{12}), d_x(2 \cdot k_{22}), d_x(k_{12} + 2 \cdot k_{22})\}$ . These different values are assigned generalised logical parameters  $\{K_{1,}, K_{1,2}\}$  for the  $X_a$  and  $\{K_{2,}, K_{2,1}, K_{2,2}, K_{2,12}\}$  for the expression  $X_b$ . Following additional constraints are valid on the parameter  $K$ 's:  $K_{1,}, K_{1,2} \in \{0, 1\}$  with  $K_{1,} \leq K_{1,2}$ ;  $K_{2,}, K_{2,1}, K_{2,2}, K_{2,12} \in 0, 1, 2$  with  $K_{2,} \leq K_{2,1} \leq K_{2,12}$  and  $K_{2,} \leq K_{2,2} \leq K_{2,12}$ . A set values can be assigned to the parameter  $K$ 's such that the above constraints are satisfied. Additional biological reasoning, such as two transcription factors have to be simultaneously present to activate a node, can be taken into consideration while selecting the parameter  $K$ 's. The truth table representing the image of all possible states of the GRN is given in Table 2.2. Depending upon the chosen parameter  $K$ 's, the image values in the truth Table 2.2 can be different. ■



$x_a$	$x_b$	$X_a$	$X_b$
0	0	1	0
0	1	0	0
0	2	0	2
1	0	1	1
1	1	0	1
1	2	0	2

**Table 2.3:** An instance of the State transition table for the GRN in Figure 2.14 with the logical parameters specified in Example 2.5.3.

The *characteristic state* of a feedback loop is defined as the state for which each variable of the loop is located at the threshold value, above which it is active in the loop considered.

**Example 2.5.2** In GRN of Figure 2.14, there is one negative cycle formed by  $A$  and  $B$  and there is a positive self-loop on  $B$ . The loop characteristic states for these two cycles are given by  $\{\theta^1, \theta^1\}$  and  $\{-, \theta^2\}$  respectively. the value “-” implies that the node  $A$  can take any possible value in  $\{0, 1\}$ . Therefore, the characteristic state  $\{-, \theta^2\}$  can give rise to two characteristic states  $\{0, \theta^2\}$  and  $\{1, \theta^2\}$ . Now each of these three loop characteristic state are checked for stability. ■

It was been demonstrated by Snoussi and Thomas in 1993 [38] that:

1. Among the singular steady states of a system, only those which are loop characteristic can be steady.
2. On the other hand, if a state is loop-characteristic there exists a combination of logical parameter values  $K$ 's, for which it is steady.

**Example 2.5.3** For  $K_{1,.} = 0, K_{1,2} = 1, K_{2,.} = 0, K_{2,1} = 1, K_{2,2} = 2, K_{2,12} = 2$ , in Example 2.5.1, it can be shown that the loop characteristic states  $\{\theta^1, \theta^1\}$  and  $\{0, \theta^2\}$  are stable while the loop characteristic state  $\{1, \theta^2\}$  is not. The state  $\{\theta^1, \theta^1\}$  belongs to a negative cycle and hence presence of this stable loop characteristic state indicates the presence of an attractor in the state space. One can verify from the truth table in 2.3 that an attractor is indeed formed by the states  $00 \rightarrow 10 \rightarrow 11 \rightarrow 01$ . On the other hand, the  $\{0, \theta^2\}$  is part of a positive feedback loop and implies the presence of multi-stationarity (i.e. multiple steady states). Indeed there are two steady states in Table 2.3, one formed by the attractor and the other formed by a regular steady state 02. ■

With the above properties of loop characteristic states, it is no longer required to enumerate all states in the system. One just need to enumerate all the cycles in a GRN and check if the characteristic state of every cycle

(and the combination of non-overlapping cycles) is a steady state. Hence, GLA requires identification of all possible feedback loops (both positive and negative) in a GRN. Enumerating all cycles in a graph can quickly run into exponential complexity. Moreover, once a stable loop characteristic state has been identified, it has to be expanded into an actual attractor. This will require testing all the possible neighbours of the threshold values in the characteristic state. For a large GRN, this enumeration of neighbouring states can quickly run into exponential complexity, specially under the asynchronous transition model where multiple next states can exist for a given state of the network.

### 2.5.2 Implicit enumeration based approaches

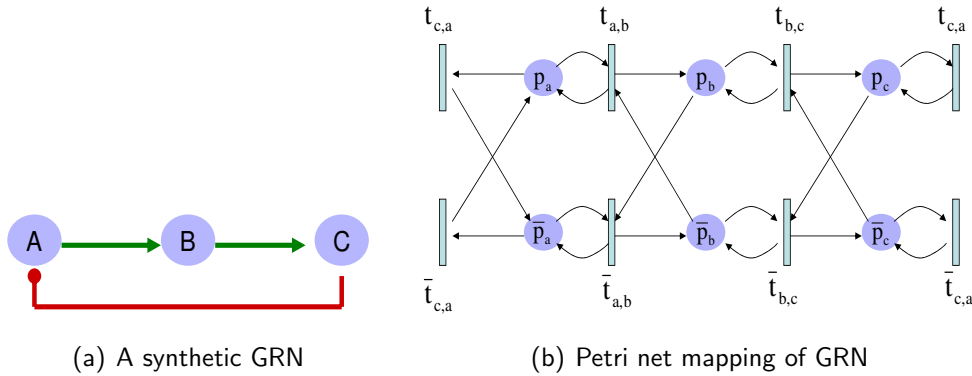
To avoid explicit enumeration of all cycles in a GRN, GLA was formalised as a *constraint satisfaction problem* by Devloo et al [148], for the first time. In this formalism, all the steady states of a system are found by solving a system of logical equations. Constraint programming is used to solve the discrete equations. The method described by Devloo allows for all steady states to be found in much larger system that was previously possible. Systems with hundreds or even thousands of components can be solved in reasonable amounts of time. However, the method does not scale well for GRNs with high in-degree and out-degree nodes. Further, the complexity involved in the expansion of characteristic steady states into real attractors still remains an issue in this approach.

A BDD-based approach has been proposed for identifying all steady states using the GLA formalisation. GINSim extends the *multiple terminal BDDs* (MTBDDs) to GLA approach [119]. However, this approach can only identify stable steady states and can not identify the cyclic attractors. Further, GLA formalism, in general, is not suitable for modeling gene perturbation experiments and the stochasticity in Boolean functions.

Another methodology based on model checking was proposed in the recent past under the modeling toolbox BioCHAM [117]. BioCHAM can perform both Boolean and continuous (kinetic models) simulation of GRNs. For Boolean simulations, BioCHAM uses a BDD based toolbox called NuSMV. In BioCHAM, given an initial state, it can compute the steady state reachable from that state. As a result, to enumerate all steady states, initialisation from all possible initial states is therefore necessary. Further, BioCHAM (and NuSMV) does not have any support for *probabilistic Boolean networks* (PBNs) and stochastic GRNs.

### 2.5.3 Petri Nets

An alternative formulation of Boolean networks has been proposed in terms of Petri Nets [36]. Before the work in [36], Petri nets were used quite extensively for modeling biochemical networks [151, 129, 110]. A Petri net representation for a small GRN is shown in Figure 2.15. A Petri net has a set of *places* and



**Figure 2.15:** (a) A synthetic GRN. (b) Petri net mapping for Boolean model corresponding to GRN.

a set of *transitions* (represented by  $p$ 's and  $t$ 's respectively. Set of tokens in places  $p$ 's represent a state of the GRN. For example, if there are tokens in places  $\{p_a, p_b, p_c\}$  then it represents the state 101. The transitions can fire asynchronously when all the incoming edges to the transition have a token. When a transition is fired, a token is transferred from all places connected to the incoming edges to places attached to the outgoing edges.

Petri nets are more suitable for biochemical simulations as reactants and metabolites consumption in chemical processes has a direct correspondence with the transition of tokens in Petri nets. However, for Boolean modeling, regulators are usually not consumed. Regulators have only activation and inhibition states and a node once activated stays in active state until an inhibiting edge is acting on it. With this extra constrain, Petri nets were extended to Boolean and multi-valued models of GRNs for the first time in [36]. However, Petri nets based computational tools for Boolean modeling of GRNs started becoming available only with the toolbox GNaPN [106]. Petri net simulations for steady state computations is a topic of research in itself for asynchronous digital circuit synthesis and simulations [8, 5] and algorithms using ROBDDs have been proposed in the context of verification of Petri net representation of asynchronous circuits (including identification of steady states) [120].

### 2.5.4 Alternate formulation of GRNs in this thesis

In this thesis we look at an alternate formulation of GRNs based on Boolean algebra and FSMs. Rather than performing analysis in terms of feedback loops in a GRN, which may not be very intuitive in the presence of multiple overlapping feedback loops, the proposed formalism models the dynamics of a GRN as a single FSM. The GRNs are mapped directly onto their corresponding Boolean function representation and the underlying Boolean state space is implicitly represented and traversed using ROBDDs. Further, the proposed formulation facilitates a common modeling framework for comput-

ing the steady states, performing gene perturbation experiments and stochastic simulations of GRNs.

---

# Boolean Modeling

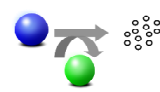
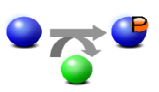
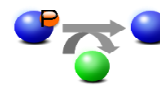

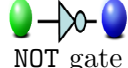
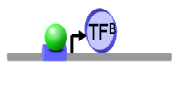
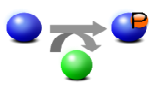
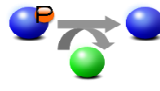


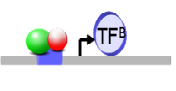
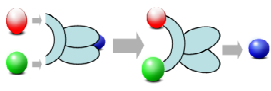
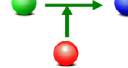

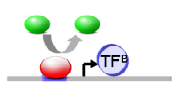

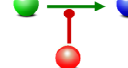
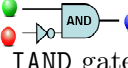


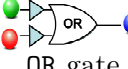
---

# 3

Availability of large amounts of experimental data suitable for inferring gene-gene or protein-protein interactions has renewed interest in Qualitative modeling of GRNs. Qualitative modeling can be used to study the dynamical behavior of a set of genes (or proteins) purely based on the information (such as degree of activation or inhibition of a gene or protein) and does not require kinetic information that is still scarcely available. Boolean modeling (where a node of a GRN can have only two activation states) and multiple-valued modeling (where nodes can take discrete levels of activation) are two such qualitative modeling approaches that are currently being pursued actively. This chapter introduces Boolean formulation of GRNs that is suitable for implicit modeling based on ROBDDs and proposes algorithms for computing steady states and perform *in silico* gene perturbation experiments on a GRN. The Boolean formulation proposed in this chapter is then extended to the multiple-valued modeling of GRNs in Chapter 4.

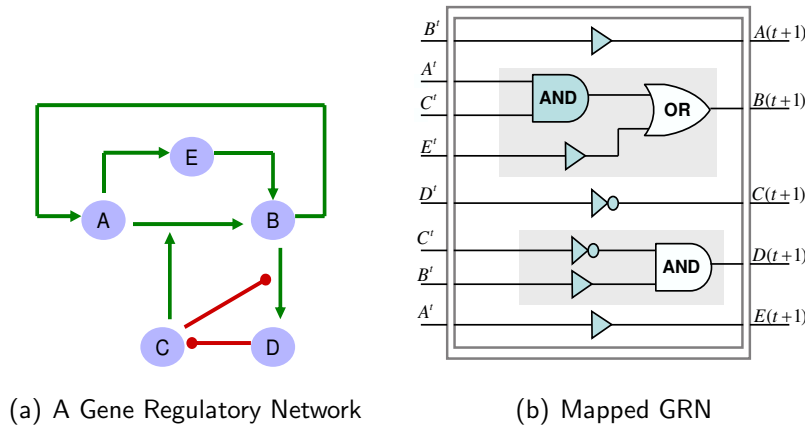
## 3.1 Boolean mapping of GRNs

Edges in a GRN represent biological phenomena underlying the activation (or inhibition) of the functional form of a gene (or protein). Almost all the underlying biological functionalities of a GRN can be represented by a combination of Boolean functions (or logic gates) in the set {AND, OR, IAND, BUFF, NOT}. Figure 3.1, gives a few instances of mapping between biological functionalities and its corresponding GRN and Boolean function representation. For example in the 1<sup>st</sup> row of Figure 3.1, the blue protein loses its activity after undergoing degradation by the green protein. This can be represented by a NOT gate which shows negative influence of green protein on the blue protein. Similarly, when a protein undergoes phosphorylation or de-phosphorylation on

Biological phenomena			Boolean function
 Protein disintegration	 Phosphorylation	 De-phosphorylation	 Inhibition  NOT gate
 Transcription	 Phosphorylation	 De-phosphorylation	 Activate  BUFF gate
 Promoter assisted transcription	 Scaffolding complex		 Assisted activation  AND gate
 Blocked transcription	 Blocked phosphorylation		 Blocked activation  IAND gate
 Dual phosphorylation sites			 Multiple activation  OR gate

**Figure 3.1:** Boolean function mapping of biological phenomena in GRNs.

interacting with another protein, it can either lose its activity (1<sup>st</sup> row) or gain its activity (2<sup>nd</sup> row). Hence, phosphorylation and de-phosphorylation are categorised under both the NOT gate and the BUFF gate (which stands for buffer). Scaffolding and a promoter-assisted gene transcription processes (in the 3<sup>rd</sup> row), requires assembling of two or more proteins into multimolecular complexes. This can be represented by an AND gate requiring simultaneous presence of two or more inputs. If the presence of a protein has an inhibiting

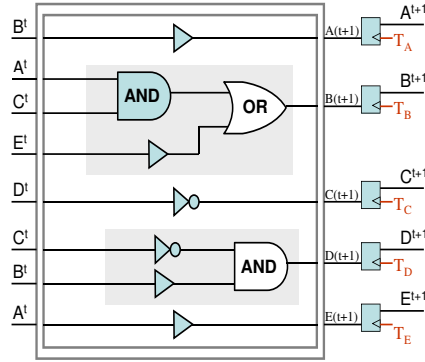


**Figure 3.2:** (a) A Gene Regulatory Network (GRN). (b) The GRN mapped to Boolean functions (gates).

effect on another protein-protein interaction (e.g. phosphorylation blocked by the presence of red protein in the 4<sup>th</sup> row) or gene-protein interaction (e.g. transcription blocked by the red protein), then it can be represented by an IAND gate. Finally, Boolean function OR represents a choice between alternate biological mechanisms to activate or deactivate a protein. For example in the 5<sup>th</sup> row blue protein can be phosphorylated at two different sites by two different proteins. Phosphorylation at any site can activate the blue protein. This phenomena is captured by the OR gate. However, the OR gate in itself does not have any biological meaning. The dual phosphorylation sites are captured by two BUFF gates in the 5<sup>th</sup> row of Figure 3.1. It should be noted that the logic gates {AND, OR, IAND} can have two or more inputs while {BUFF, NOT} are only single input functions. With the mapping defined in Figure 3.1, a synthetic GRN in Figure 3.2(a) can be translated into a logic gate representation as in Figure 3.2(b).

### 3.1.1 Network states and steady states

In Boolean modeling of GRNs, a node can have only two expression states: *active* and *inactive*. These expression states of a node are represented by 1 and 0 respectively. A snapshot of the activity level of all the nodes in the GRN at time  $t$  is called the *state of the network*. The network can transition from one state to another state as defined by the Boolean functions. Hence, a state of the network evolves over time by making a transition into another state until it stabilises into an *attractor* (or the steady state). Such an attractor represents the long term behaviour of the genes/proteins in the regulatory networks. Under the Boolean assumption, if there are  $N$  genes in a GRN, the network can be in any of the  $2^N$  possible states. This exponential state space makes the identification of attractors both a computationally and a memory intensive task.



**Figure 3.3:** Mapped GRN with registers showing the delay in expression of each output node.

In biology, a state of the network corresponds to the measured activity of all the genes (or proteins) in a cell at a given time instant and can be experimentally measured using techniques such as flow-cytometry or microarray. The state transition of a GRN biologically corresponds to the dynamic evolution of the cell. With the existing experimental techniques it is difficult to monitor all the state transitions inside a cell. However, steady state behavior, which corresponds to the end point of an experiment when all cells stabilise, is easier to measure experimentally. Attractors (or the steady states) of Boolean models of GRNs have been shown to correspond to the cellular steady states (or cellular phenotypes) in the past [143, 139].

### 3.1.2 Synchronous vs. Asynchronous:

In Figure 3.2(b), based on the activity of the nodes at the input at time  $t$  (represented by  $A^t, B^t, \dots, E^t$ ) and the corresponding Boolean functions for every node, the activity of the node at the next time instance is given by functions  $A(t+1), B(t+1)$  and so on. The underlying biological functions for every node in the GRN will have a latency between the time input signal is sensed and the time when the change in expression is produced at the output. This latency of biological processes can be mapped to the latency of Boolean functions in Figure 3.2(b). However, unlike in digital circuits where a logic gate has a unique latency, in GRNs the same logic gate can have different latency for different nodes depending upon factors such as length of protein coding regions for the transcription process and location of phosphorylation site. Further it is hard to experimentally measure the latency of each biological function. In the absence of such an information, it is reasonable to model the latency in the transition of expression of a node from 0-to-1 or vice versa rather than the latency in each logic gate. Latency in the transition of expression can be introduced by adding a register at each output node (see Figure 3.3). The register stores the value of Boolean functions (e.g.  $A(t+1), B(t+1)$  etc) and



updates the stored value with the new value at the time intervals given by  $T_A$ ,  $T_B$ , and so on for every node in the GRN.

Delay in transition introduces a notion of synchronicity and asynchronicity in the dynamics of the GRN. If all the nodes update the value stored in their output registers at the same time instance (i.e.  $T_A = T_B = \dots = T_E$ ) then the system is said to behave synchronously. Otherwise, the system behaves asynchronously. Even though in theory register update times can be inferred from kinetic rate parameters, this information is seldom available in practice. To address this problem, the asynchronous models can be relaxed to fully asynchronous model wherein all registers are assumed to have a different update times (i.e.  $T_A \neq T_B \neq \dots \neq T_E$ ). This assumption of asynchronous models can also be viewed as frequent measurements such that only one gene (or protein) can update its expression between two measurement steps. The assumption is also biologically plausible as two unrelated biological events can rarely occur at the exact same time instant. The relaxed formulation of asynchronous models has been used quite often in the literature [131, 132]. Next section formally introduces the synchronous and asynchronous Boolean modeling of GRNs.

## 3.2 Problem formulation

Given a GRN (such as in Figure 3.2(a)), expression of a node (or gene)  $i$  at time  $t$  is represented by a Boolean variable  $x_i^t$ . If  $x_i^t = 0$ , the node  $i$  is inactive and if  $x_i^t = 1$ , the node is active. The state of the network at time  $t$  is represented by a Boolean vector,  $\mathbf{x}^t$ , of size  $N$  (number of genes in the network) and is referred to as a *present state* vector. Each bit of this vector (represented by  $x_i^t$ ) represents whether the gene is active or inactive. Another Boolean vector,  $\mathbf{x}^{t+1}$ , of size  $N$  is used to represent the state of the network in the next step and is called the *next state* vector. The expression of each gene  $i$  at time  $t + 1$  can be written as a function  $x_i(t + 1)$  of the state of the genes acting as its input at time  $t$ . Equations 3.1 and 3.2 can be used to compute the function  $x_i(t + 1)$  and can be understood better with the help of Figure 3.2. In Equation 3.1, an activator (or an inhibitor) function  $f_{x_i}^{ac}(t)$  (or  $f_{x_i}^{in}(t)$ ), represent the set of genes that have a collective activating (or inhibiting) impact on the gene  $i$ . Equations 3.1 and 3.2 can be also formed by composing Boolean gates as in Figure 3.2(b).

$$x_i(t + 1) = \left( \bigvee_{l=1}^n f_{x_{i,l}}^{ac}(t) \right) \wedge \neg \left( \bigvee_{l=1}^n f_{x_{i,l}}^{in}(t) \right) \quad (3.1)$$

$$f_{x_{i,l}}^{ac,in}(t) = \left( \bigwedge_{j=1}^p x_j^{t,ac} \right) \wedge \left( \bigwedge_{j=1}^p \neg x_j^{t,in} \right) \quad (3.2)$$

$$x_j \in \{0, 1\}$$

$f_{x_m}^{ac}$  are the set of activator functions of  $x_i$

$f_{x_n}^{in}$  are the set of inhibitor functions of  $x_i$

$x_p^{ac}$  are the set of activators of functions  $f_{x_i}$

$x_q^{in}$  are the set of inhibitors of functions  $f_{x_i}$

$\wedge$ ,  $\vee$  and  $\neg$  represent Boolean AND, OR and NOT

**Example 3.2.1** For the synthetic GRN in Figure 3.2, Boolean functions describing the expression of nodes are given by the following equations :

$$\begin{aligned} x_A(t+1) &= x_B^t \\ x_B(t+1) &= (x_A^t \wedge x_C^t) \vee x_E^t \\ x_C(t+1) &= \neg x_D^t \\ x_D(t+1) &= x_B^t \wedge \neg x_C^t \\ x_E(t+1) &= x_A^t \end{aligned}$$

Boolean variables  $x_A^t$ ,  $x_B^t$ ,  $x_C^t$  and  $x_D^t$  represent the expression of nodes  $A$ ,  $B$ ,  $C$  and  $D$  respectively. ■

Equations 3.1 and 3.2 represent the dynamics of individual genes independent of the dynamics of the other genes in the network. To model the dynamics of the complete network, one has to couple the dynamics of these genes. This can be done by defining the transition function,  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$ , of the state of the network. Function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  represents the transition from the present state  $\mathbf{x}^t$  to the next state  $\mathbf{x}^{t+1}$ . Transition functions  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$ , can be different depending upon synchronous or asynchronous dynamics model (as will be described in the next section).

### 3.2.1 Synchronous model

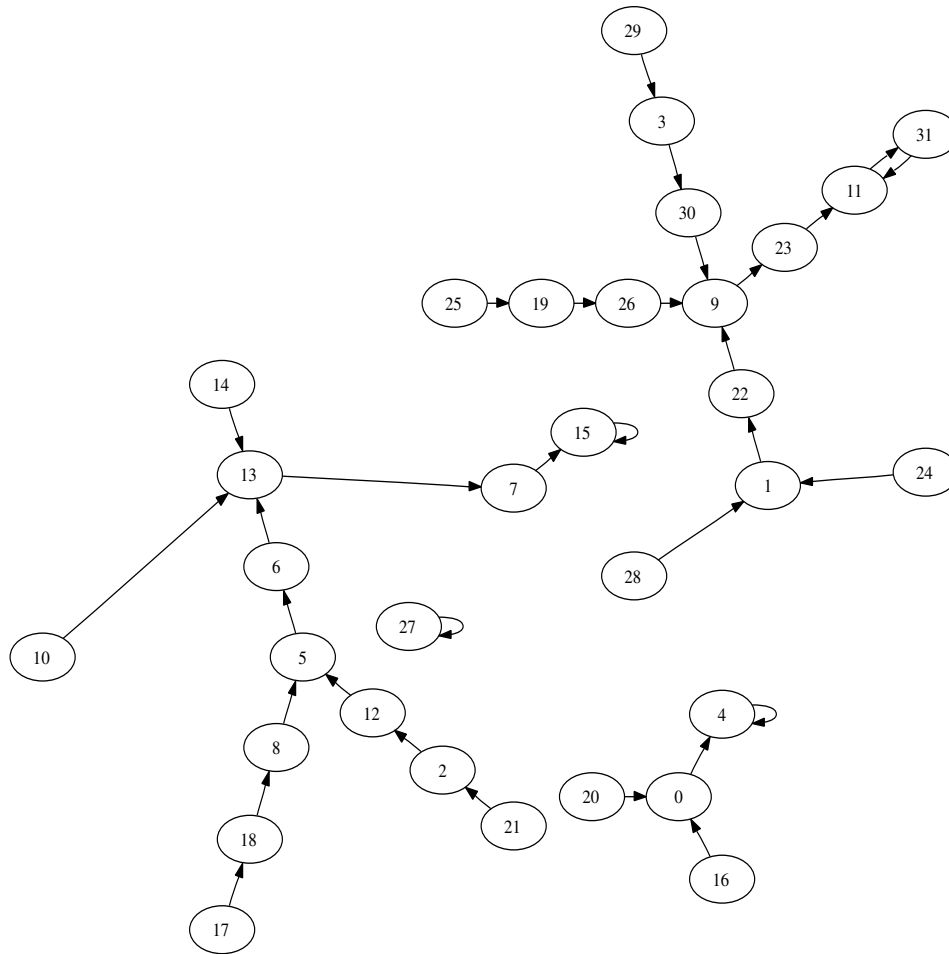
If the transition function is synchronous, expression of all genes is updated at the same time. A synchronous model can be described by the following set of equations :

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = x_i^{t+1} \leftrightarrow x_i(t+1) \quad (3.3)$$

$$T(\mathbf{x}^t, \mathbf{x}^{t+1}) = T_0(\mathbf{x}^t, \mathbf{x}^{t+1}) \wedge \dots \wedge T_N(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (3.4)$$

Equation 3.3 gives the transition function for a single gene  $i$ . Symbol  $\leftrightarrow$  stands for logical equivalence and in Equation 3.3, represents that the value of a gene in the next time step,  $x_i^{t+1}$  is equal to the value of the function  $x_i(t+1)$  (as defined in Equation 3.1). Equation 3.4 states that all genes in the network make a simultaneous transition from the present state  $\mathbf{x}^t$  to the next state  $\mathbf{x}^{t+1}$ .

If a state transition graph is constructed using Equations 3.3 and 3.4, then each state has only one transition going out of it. Hence, assuming that all



**Figure 3.4:** State Transition Diagram for the synchronous model of the GRN in Figure 3.2.

genes can take ‘0’ or ‘1’ levels of expression, the number of states and the number of transitions in the state transition graph are both equal to  $2^N$ , where  $N$  is the number of genes in the network. The state transition diagram for the synthetic GRN in Figure 3.2(a) is shown in Figure 3.4. The  $N$  bit state vector  $\mathbf{x}^t$  is packed into an integer in Figure 3.4 for aesthetical reasons.

### 3.2.2 Asynchronous model

In the asynchronous model, no two genes (or proteins) can change their expression states simultaneously. More precisely, following three assumptions are made: (a) Only one gene can make a transition (or be updated) in a single step. (b) At least one gene makes a transition unless none of the genes can change their expression levels (i.e.  $x_i(t+1) = x_i^t \forall i$ ). (c) Every gene is equally likely to make a transition. That means every state can have potentially  $N$  successor states, where each successor state differs from the present state in

only one gene expression.

The first assumption of asynchronous models, which states only one gene can be updated between two consecutive time steps, can be specified by using Equation 3.5:

$$TP_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = (x_i^{t+1} \leftrightarrow x_i(t+1)) \wedge \bigwedge_{j \neq i} (x_j^{t+1} \leftrightarrow x_j^t) \quad (3.5)$$

Equation 3.5 states that gene  $i$  in the next time step, takes the value as defined by the function  $x_i(t+1)$  in Equation 3.1 and all other genes stay at their current expression levels. For the second assumption, one has to check if all genes stay at the same expression level in the next time step. This can be specified by using the *flag function*,  $F(\mathbf{x}^t)$  defined in Equation 3.6.

$$F(\mathbf{x}^t) = \bigwedge_{i=1}^N (x_i(t+1) \oplus x_i^t) \quad (3.6)$$

The Flag function  $F(\mathbf{x}^t) = 0$  iff  $x_i(t+1) \neq x_i^t$  for at least one gene  $i$ . The transition relation for gene  $i$  is then given by Equation 3.7.

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \{F(\mathbf{x}^t) \vee (x_i(t+1) \oplus x_i^t)\} \wedge TP_i(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (3.7)$$

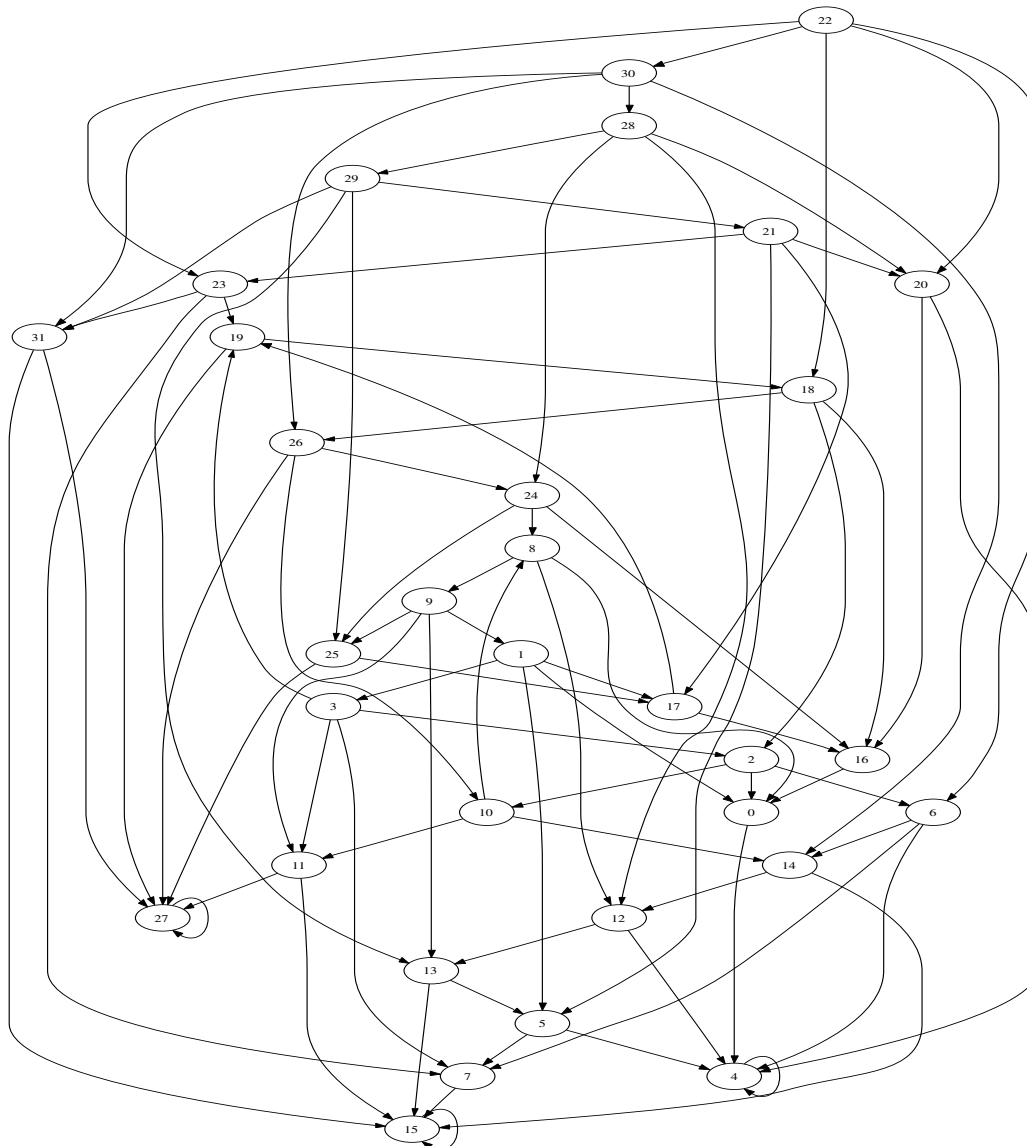
In Equation 3.7, the transition function for gene  $i$  is given by  $TP_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  if either all the genes stay at the same level in the next time step (i.e. the flag function,  $F(\mathbf{x}^t) = 1$ ) or if the expression of gene  $i$  in the next step can be different from its expression in the present time step (i.e.  $(x_i(t+1) \oplus x_i^t) = 1$ ). Equation 3.7 is the asynchronous counterpart of Equation 3.3 of the synchronous model.

Transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  for the state of the network is then given by Equation 3.8.

$$T(\mathbf{x}^t, \mathbf{x}^{t+1}) = T_0(\mathbf{x}^t, \mathbf{x}^{t+1}) \vee \dots \vee T_N(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (3.8)$$

Equation 3.8 incorporates the third assumption, which states that from a given state, the network can have multiple next states and each next state can differ from the present state in at most one gene expression. Note that Equation 3.8 is the counterpart of Equation 3.4 for synchronous models, with the difference that the conjunctions ( $\wedge$ ) have been replaced by disjunctions ( $\vee$ ).

The asynchronous model has  $2^N$  states, and each of them can have up to  $N$  outgoing transitions, making a total of  $N \cdot 2^N$  transitions in the worst case. This means that the number of transitions in an asynchronous model can be more than those in the corresponding synchronous model by Upton a factor of  $N$ . The asynchronous state transition diagram for the synthetic GRN of Figure 3.2(a) is shown in Figure 3.5. At this point, one should appreciate the complexity of transitions in an asynchronous model as compared to a synchronous model of the same GRN. This increased number of transitions, as will be seen in Section 3.2.6, can have a considerable impact on the computation time of the algorithm.



**Figure 3.5:** State Transition Diagram for the asynchronous model.

### 3.2.3 Boolean attractors

Boolean attractors and steady states of a GRN can be described formally by the following set of definitions :

**Definition 6** *Successor.* Given a state of the network  $\mathbf{x}^t$ , all the states  $\tilde{\mathbf{x}}^t$  such that  $T(\mathbf{x}^t, \tilde{\mathbf{x}}^t) = 1$  are the successor states of the state  $\mathbf{x}^t$ .

**Definition 7** *Predecessor.* Given a state of the network  $\mathbf{x}^t$ , all the states  $\tilde{\mathbf{x}}^t$  such that  $T(\tilde{\mathbf{x}}^t, \mathbf{x}^t) = 1$  are the predecessor states of the state  $\mathbf{x}^t$ .

**Definition 8** *Forward image.* Given a set of states  $S(\mathbf{x}^t)$ , the forward image  $I_T^f(S(\mathbf{x}^t))$  is the set of immediate successors of the states in the set  $S(\mathbf{x}^t)$  under the state transition graph defined by the transition function  $T$ .

**Definition 9** *Backward image.* Given a set of states  $S(\mathbf{x}^t)$ , the backward image  $I_T^b(S(\mathbf{x}^t))$  is the set of immediate predecessors of the states in the set  $S(\mathbf{x}^t)$  under the state transition graph defined by the transition function  $T$ .

**Definition 10** *Forward reachable states.* Given a set of states  $S^0$ , forward reachable states  $FR(S^0)$  are the set of states that can be reached from the states in the set  $S_0$  by iteratively computing forward image under the transition function  $T$  until no new states are reachable.

**Definition 11** *Backward reachable states.* Given a set of states  $S^0$ , backward reachable states  $BR(S^0)$ , are all the states  $\mathbf{x}^t$  whose forward reachable set contain at least one state in  $S_0$ .

**Definition 12** *Attractor.* An attractor is a set of states  $SS(\mathbf{x}^t)$  such that for all the states  $s \in SS(\mathbf{x}^t)$ , the forward reachable set  $FR(s)$  is the same as  $SS(\mathbf{x}^t)$  (i.e.  $FR(s) = SS(\mathbf{x}^t) \forall s \in SS(\mathbf{x}^t)$ ).

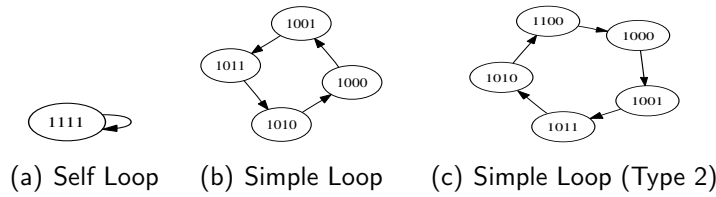
**Definition 13** *A Steady State is an attractor that consists of a single state.*

**Example 3.2.2** Let us assume that the system is in a state  $S_0 = 5$  in the synchronous state transition diagram of Figure 3.4. The successor of state  $S_0$  is the state 6 and the predecessors of  $S_0$  are the states 8 and 12 respectively. Set of forward reachable states from  $S_0$  are given by  $FR(S^0) = \{6, 13, 7, 15\}$  and the set of backward reachable states is given by  $BR(S^0) = \{17, 18, 8, 12, 2, 21\}$ . Since,  $FR(S^0) \neq S_0$ , the starting state  $S^0$  does not belong to an attractor.

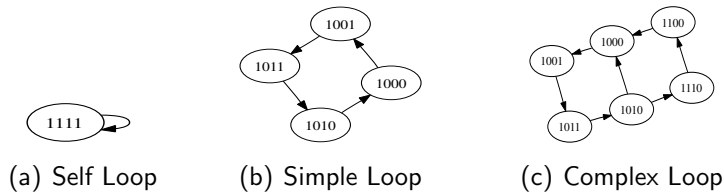
There are four attractors in Figure 3.4, given by  $SS^1 = \{4\}$ ,  $SS^2 = \{27\}$ ,  $SS^3 = \{15\}$  and  $SS^4 = \{11, 31\}$ . Out of these four attractors,  $SS^1$ ,  $SS^2$  and  $SS^3$  are the steady states. ■

It is interesting to notice the differences between the structure of attractors in a synchronous (Figure 3.6) and asynchronous model (Figure 3.7) of the same GRN. Following Definition 12, single state attractor forms a *Self Loop*, otherwise attractor can be either a *Simple Loop* or a *Complex Loop*. A *Simple Loop* is a cycle of states such that each state can have exactly one successor state. A *Complex Loop* is formed by two or more overlapping simple Loops. A self loop attractor is called a steady state (Definition 13).

Since synchronous networks can have only one outgoing transition from any state, an attractor in synchronous networks can only be of two types: a) self loops, and b) simple loops. Simple loops for synchronous models can again be divided into two subclasses : b1) loops where any two consecutive states differ in exactly a single gene expression and b2) loops where at least one pair



**Figure 3.6:** Possible types of attractors in a synchronous model



**Figure 3.7:** Possible types of attractors in an asynchronous model

of consecutive states differ in more than one gene expression (e.g. transition  $1010 \rightarrow 1100$  in Figure 3.6(c)).

Similarly, an asynchronous model can have three classes of attractors (Figure 3.7), namely: a) self loops, b) simple loops, and c) complex loops. Since the definition of asynchronous models allows only a single gene expression change between any consecutive states, they can have only one kind of simple loops (unlike synchronous models).

Self loops and simple loops found in the asynchronous model of a gene regulatory network are also present in the synchronous model of the same network. The two models can only differ in the complex loop and simple loops of the second type (i.e. Figure 3.6(c)). Even though simple loops (of Figure 3.6(c)) may lead to a complex loop (of Figure 3.7(c)) in an asynchronous model, for some gene regulatory networks, the presence of the former is not necessary for the existence of complex loops. Hence, some attractors seen in a synchronous model can vanish in the corresponding asynchronous model of a GRN. This can also be seen from the state transition diagrams of synchronous and asynchronous models of synthetic GRN in Figure 3.2(a). The synchronous steady state formed by  $\{11 \rightarrow 31 \rightarrow 11\}$  in Figure 3.4 does not exist in the asynchronous model (Figure 3.5).

Computing attractors (or steady states) of a GRN is of interest to biologists as attractors have a biological correspondence to cell states (or cell phenotypes) [143, 139, 56]. Further, since the steady state of a GRN corresponds to the end point of an experiment when all cells stabilise, it is easily to experimentally validate the steady state behavior than validating the behaviour of transient states of a GRN. In the next section, we give algorithms that can be used for identifying all the attractors in a GRN.

### 3.2.4 Algorithms for computing attractors

As explained in Sections 3.2.1 and 3.2.2, a state transition graph can have exponential state space and if this graph is explicitly represented and traversed, then an exponential number of states restricts the computation to small sized networks. Further, identifying attractors by enumeration becomes difficult as one will have to consider all possible subset of states that can form an attractor (which can be super-exponential in the worst case). To avoid explicit enumeration of subset of states, a set of theorems proposed in [7] can be used (stated again in Theorems 1 and 2).

**Theorem 1** *A state  $i \in S$  is a part of an attractor if and only if  $FR(i) \subseteq BR(i)$ . State  $i$  is transient otherwise.*

**Theorem 2** *If state  $i \in S$  is transient, then states in  $BR(i)$  are all transient. If state  $i$  is a part of an attractor, then all the states in  $FR(i)$  are also part of the same attractor. In the latter case set  $\{BR(i) - FR(i)\}$  has all the transient states.*

**Example 3.2.3** Let us assume we will like to test if the state  $S_0 = 5$  in the synchronous state transition diagram of Figure 3.4 belongs to an attractor or not. From Example 3.2.2,  $FR(S^0) = \{6, 13, 7, 15\}$  and  $BR(S^0) = \{17, 18, 8, 12, 2, 21\}$ . As we can clearly see  $FR(S^0) \not\subseteq BR(S^0)$ . Hence Theorem 1 implies that the state  $S^0$  does not belong to an attractor. Then Theorem 2 implies that none of the states in  $BR(S^0) = \{17, 18, 8, 12, 2, 21\}$  should belong to an attractor either. One, can verify from attractors listed in Example 3.2.2 that none of the states in the set  $BR(S^0)$  indeed belong to any of the four attractors. ■

Based on Theorems 1 and 2, the procedure for attractor computation is given in Algorithm 2. This algorithm takes as input the transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$ , which can be either synchronous or asynchronous. In Line 5 of Algorithm 2, a seed state is selected from the state space  $T'$  and forward and backward reachable states from this seed state are computed in Lines 6 and 7. Then Theorem 1, as implemented in Line 8, checks if the seed state (from Line 5) is part of an attractor. If the seed state is indeed part of an attractor, then using Theorem 2 (as implemented in Lines 9-12), all the states in the forward reachable set are declared to from an attractor in Line 9 and the rest of the states in the backward reachable set are declared transient in Line 10. Otherwise, the seed state and all the other states in the backward reachable set are declared transient in Line 12. In Line 13, the state space is reduced by removing those states that have already been tested for reachability and the process is repeated to find another attractor on the reduced state space. This process is iterated until the whole state space is explored (i.e. until  $T \neq \emptyset$ ). The states in the backward reachable set are removed from the state space in each iteration, resulting in the continuous size reduction of the latter. One



**Algorithm 2:** Algorithm for computing Attractors

---

```

1 all_attractors(T)
2 begin
3    $T' \leftarrow T$ 
4   while  $T' \neq \emptyset$  do
5      $s \leftarrow \text{initial\_state}(T')$ 
6      $FR(s) \leftarrow \text{forward\_set}(s, T')$ 
7      $BR(s) \leftarrow \text{backward\_set}(s, T')$ 
8     if  $FR(s) \wedge \overline{BR(s)} = \emptyset$  then
9       report  $FR(s)$  as an attractor
10      report  $BR(s) \wedge \overline{FR(s)}$  as all transient states
11    else
12      report  $s \vee BR(s)$  as all transient states
13     $T' \leftarrow T' \wedge \overline{s \vee BR(s)}$ 
14 end
15 initial_state(T)
16 begin
17    $s(V_t) = \text{random\_state}(T)$ 
18    $RS^{(0)} \leftarrow \emptyset, FS^{(0)} \leftarrow \{s\}$ 
19    $k \leftarrow 0$ 
20   while  $FS^{(k)} \neq \emptyset$  do
21      $FS^{(k+1)} = I_T^f(FS^{(k)})(\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t) \wedge \overline{RS^{(k)}}$ 
22      $RS^{(k+1)} = RS^{(k)} \vee FS^{(k+1)}$ 
23      $k \leftarrow k + 1$ 
24    $s \leftarrow \text{random\_state}(FS^{(k-1)})$ 
25   return  $s$ 
26 end

```

---

should note that the number of iterations of Lines 4-13 depends upon how the seed state is selected in Line 5.

Function *initial\_state()* in Algorithm 2 selects a seed state from the given state space  $T'$ . In this function (implemented in Lines 17-25), a random initial state is selected from the transition state space  $T$  in Line 17. The forward reachable set from this random initial state is then computed in Lines 19-24. During the forward set computation, when the frontier set evaluates to  $\emptyset$  in iteration  $k$ , a random state is taken from the frontier set in iteration  $k - 1$  and returned as the seed state. The motivation behind this function is that a state in the last frontier set is more likely to be a part of an attractor than a random state in the state space  $T$ . For synchronous models, it can be proved that the seed state selected in this way is guaranteed to be a part of an attractor. While for the asynchronous models, there is no guarantee that the seed state is part of an attractor.

Algorithm 2 also uses functions *forward\_set()* and *backward\_set()* for computing forward reachable  $FR(S)$ , and backward reachable  $BR(S)$  states, respectively. These functions are given in Algorithm 3. In Algorithm 3, the while

**Algorithm 3:** Computing Forward and Backward reachable sets

---

```

1 forward_set( $S^0, T$ )
2 /* backward_set( $S^0, T$ ) */
3 begin
4    $RS^{(0)} \leftarrow \emptyset, FS^{(0)} \leftarrow \{S^0\}$ 
5    $k \leftarrow 0$ 
6   while  $FS^{(k)} \neq \emptyset$  do
7      $FS^{(k+1)} = I_T^f(FS^{(k)})(x^{t+1} \leftarrow x^t) \wedge \overline{RS^{(k)}}$ 
8     /*  $FS^{(k+1)} = I_T^b(FS^{(k)})(x^t \leftarrow x^{t+1}) \wedge \overline{RS^{(k)}}$  */
9      $RS^{(k+1)} = RS^{(k)} \vee FS^{(k+1)}$ 
10     $k \leftarrow k + 1$ 
11  return ( $FR(S^0) \leftarrow RS^{(k)}$ )
12  /* return ( $BR(S^0) \leftarrow RS^{(k)}(x^{t+1} \leftarrow x^t)$ ) */
13 end

```

---

loop in Lines 6-10 computes the reachable states iteratively starting from the initial set of states  $S^0$ , where  $k^{th}$  iteration represents the states reachable in  $k$  time steps from  $S^0$ .  $FS^k$  and  $RS^k$ , are the frontier set and the reachable set respectively in the  $k^{th}$  iteration of the while loop. The Frontier set in the  $k^{th}$  iteration, contains the states which have been reached for the first time in the  $(k - 1)^{th}$  iteration of the while loop. The Reachable set in the  $k^{th}$  iteration contains all reachable states from the initial set  $S^0$  up to  $k$  iterations. The Frontier set in iteration  $k + 1$  is computed by taking the forward image (backward image for backward reachable set computation) of the frontier set in the  $k^{th}$  iteration and removing from this image set, the states that have already been explored in previous iterations (which are stored in Reached Set). The Reached Set is updated by adding the new states from the frontier set. This process is iterated until no new states can be added to Reached Set. The final Reached Set represents the forward (backward) reachable set from the set of initial states  $S^0$ .

### 3.2.5 Algorithm complexity

Given a graph  $G(V, E)$ , identifying all the attractors in  $G$  can be performed in linear time  $O(|V| + |E|)$  by depth-first search algorithm. However, for Boolean functions there can be exponential number of vertices (i.e.  $|V| = 2^N$ ) in the graph for  $N$  Boolean variables, resulting in an exponential space complexity of the problem of identification of all the attractors in the Boolean state space. BDDs can be used to represent the Boolean state space within a reasonable memory requirements and perform efficient reachability analysis on the underlying graph. However, it is difficult to analyze the complexity of BDD based algorithm which further depends upon the size of the BDD representation of the problem. BDD representation can be exponential with the problem size in the worst case scenario, but in the most practical cases it has a mild growth with the size of the problem as has been demonstrated in the

Network	Nodes	Edges	Number of Attractors				Time taken (in sec)	
			Self	Simple	Simple 2	Complex	sync	async
Mammalian	10	39	1	0	1	1	0.1	0.26
T-helper	23	34	3	0	0	0	0.12	0.35
Dendritic	114	129	0	1	0	0	0.32	0.37
T-cell receptor	40	58	1	0	9	7	3.0	960
Network 1	1263	5031	1	0	0	0	200	*

**Table 3.1:** Benchmarking of the synchronous model (column 8) using Algorithm 2 and asynchronous model (column 9) using Algorithm 2. A cut-off time of 1 hour was used and the algorithms which could not finish computation within this time were terminated (represented by '\*'). Mammalian Cell Network is taken from [3], T-helper from [102] and T-cell receptor from [140]. The Dendritic Cell network was generated by semi-automatic mining of literature evidence. Network 1 is a full literature mined Insulin Growth Factor regulatory network. It has been developed through automatic literature mining tools that build a tentative regulatory network based on the set of keywords such as activation/inhibition.

literature [133, 63, 46]. Therefore in the worst case scenario, the complexity of BDD based Algorithm 2 can be the same as an explicit depth-search algorithm. But, as we will see in the next section, it runs very efficiently for most of the big GRNs.

### 3.2.6 Computational results

Run times of Algorithm 2 on some of the benchmark networks are given in Table 3.1. From the results it can be seen that the synchronous algorithm scales well with the size of the network and can compute all the attractors in reasonable time and memory. The benchmarking was performed on a 1.8 GHz Dual Core Pentium machine with 1 GB of RAM running on Linux Fedora Core 5.

The last column of Table 3.1 presents the benchmarking of the asynchronous model. The increased run time to compute the attractors in asynchronous models is due to both the size of the state transition diagram and the heuristics used to select seed states in Algorithm 2. Contrary to synchronous models, for the asynchronous model, the *initial\_state()* function in Algorithm 2 does not guarantee to return a state that forms a part of an attractor. This creates a potential problem, since, for computing the set of attractors common to the synchronous and the asynchronous model of a GRN, Algorithm 2 for the asynchronous model may require a large number of iterations. For this reason, in Table 3.1, Mammalian Cell, T-helper and T-cell receptor have a greater differences in computational time for synchronous and asynchronous model as compared to the run time difference for the Dendritic Cell network. The large time difference between the two models for the Network 1 is due to the extremely large size and complex configuration of this network.

### 3.3 Modeling asynchronicity using a synchronous Model

As is evident from the results from Table 3.1, BDD representation and manipulation can be far more efficient on a synchronous representation as compared to the asynchronous representation. A similar conclusion has been made in an entirely different context in *electronic design automation* (EDA) community, where verification of asynchronous circuits using BDDs is known to be more computationally challenging than synchronous circuits [80, 108]. However, by taking into consideration the similarities in the structure of attractors of synchronous and asynchronous models, run times of asynchronous models of GRN in Table 3.1 can be improved. As explained in Section 3.2.3, synchronous and asynchronous models only differ in attractors formed by complex loops and a specific type of simple loops. To reduce the computation times, algorithms given in Section 3.2.4 can be used to compute the common attractors on the synchronous model and then compute the complex loop attractors on the asynchronous model. This can improve the efficiency of the algorithms proposed in Section 3.2.4 for two reasons :

1. The common set of attractors can be computed in fewer iterations of Algorithm 2 for the synchronous models than that required for the asynchronous model.
2. ROBDDs for asynchronous models are more complex than those for synchronous models. This makes all the logic operations like AND, OR, Quantify, etc. computationally demanding. Computing some of the attractors on the synchronous model should improve the computational efficiency.

#### 3.3.1 Combined synchronous-asynchronous algorithm

Algorithm 4 details the combined synchronous-asynchronous traversal technique. In this algorithm, synchronous attractors are first computed in line 3. Then in lines 4-7, the synchronous attractors that do not exist in the asynchronous model are deleted. In line 8, the backward reachable states from the remaining attractors are computed on the asynchronous state transition diagram. These backward reachable states are removed from the state space in line 9 (using Theorem 2) and the remaining attractors of the asynchronous model are computed in line 10 on the reduced state space using Algorithm 2.

The function *isFalseLoop()* in lines 12-28 checks for the false synchronous attractors. In line 14 of this function, a state  $s_0$  is randomly selected from the states set  $S$ . Then two sets, namely the reached set  $RS^0$  and frontier set  $FS^0$ , are defined and initialised to null and  $s_0$  respectively. The superscript of  $FS$  and  $RS$  stands for the iteration number. Then the state reachable in one step from the current frontier set is computed in line 18. Since the reachable states are computed on synchronous models, there would be only one state in

**Algorithm 4:** Computing asynchronous attractors

---

```

1  comp_async_attractors( $T_{sync}, T_{async}$ )
2  begin
3  |   $SS_{sync}[] = all\_attractors(T_{sync})$ 
4  |  for  $i = 0$  to  $SS_{sync}.size()$  do
5  |  |  if isFalseLoop( $SS[i], T_{sync}$ ) == false then
6  |  |  |   $SS_{async} = SS_{async} \cup SS_{sync}[i]$ 
7  |  |  |  report  $SS[i]$  as an attractor
8  |   $BR(SS_{async}) \leftarrow backward\_set(SS_{async}, T_{async})$ 
9  |   $T'_{async} \leftarrow T_{async} \wedge \overline{SS_{async} \vee BR(SS_{async})}$ 
10 |   $SS_{async}[] = all\_attractors(T'_{async})$ 
11 end

12 isFalseLoop( $S, T$ )
13 begin
14 |   $s_0 = random\_state(S)$ 
15 |   $RS^{(0)} \leftarrow \emptyset, FS^{(0)} \leftarrow \{s_0\}$ 
16 |   $k \leftarrow 0$ 
17 |  while  $FS^{(k)} \neq \emptyset$  do
18 |  |   $FS^{(k+1)} = I_T^f(FS^{(k)})(x^{t+1} \leftarrow x^t)$ 
19 |  |   $nVarDiff = \sum_{i=1}^N FS_i^{(k)} \oplus FS_i^{(k+1)}$ 
20 |  |  if  $nVarDiff \geq 2$  then
21 |  |  |  /* false asynchronous attractor */
22 |  |  |  return true
23 |  |   $FS^{(k+1)} = FS^{(k+1)} \wedge \overline{RS^{(k)}}$ 
24 |  |   $RS^{(k+1)} = RS^{(k)} \vee FS^{(k+1)}$ 
25 |  |   $k \leftarrow k + 1$ 
26 |  /* genuine asynchronous attractor */
27 |  return false
28 end

```

---

the new frontier set. Then in line 19, the number of bits by which the current and the new frontier set differ is computed. If the number of bits by which these two states differ is more than one, then the attractor is declared false (lines 20-22). Otherwise, the new frontier set is modified in line 23. If the new frontier state has already been explored then this modification makes it an empty set. The new frontier set is added to the reached states set (line 24) and the process is iterated until the the frontier set is empty (line 17). If for all the consecutive states of an attractor, the number of bits by which they differ is exactly one, then the attractor is declared genuine (line 27).

### 3.3.2 Computational results

Results obtained on using this combined model are listed in the last column of Table 3.2. The improvement of the combined model over the asynchronous model is more evident from the results on T-cell receptor and Network 1 gene

Network	Nodes	Edges	Number of Attractors				Time taken (in sec)		
			Self	Simple	Simple 2	Complex	sync	async	combined
Mammalian	10	39	1	0	1	1	0.1	0.26	0.22
T-helper	23	34	3	0	0	0	0.12	0.35	0.4
Dendritic	114	129	0	1	0	0	0.32	0.37	0.49
T-cell receptor	40	58	1	0	9	7	3.0	960	460
Network 1	1263	5031	1	0	0	0	200	*	730

**Table 3.2:** Benchmarking of the combined synchronous asynchronous model (column 10) vs. stand-alone synchronous and asynchronous models. Network descriptions are as in Figure 3.1.

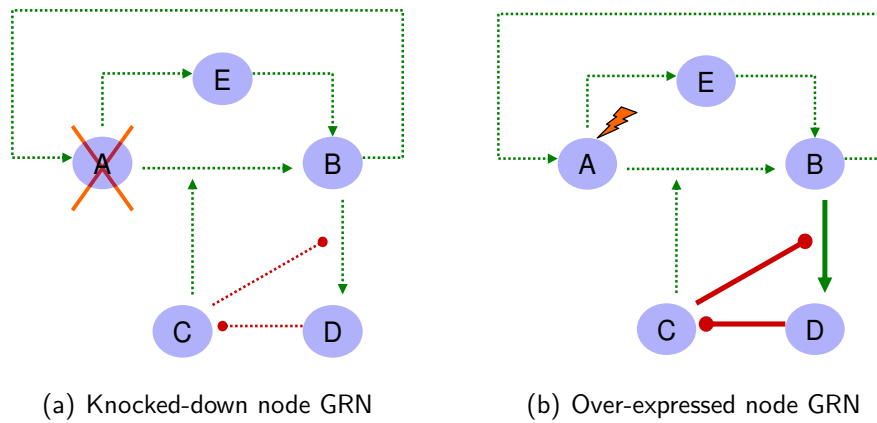
regulatory networks. While processing for Network 1, the asynchronous model could not finish the computation in 1 hour whereas the combined method computed the attractors in 12 minutes and for the T-cell receptor network, the performance almost doubled. For T-helper and Dendritic Cell networks, the combined model takes marginally more time than the asynchronous model but this might be attributed to the fact that there is a fixed overhead involved in computing the backward reachable set in Line 8 of Algorithm 2 that is not compensated by the small run time difference between the synchronous and the asynchronous model.

### 3.4 Modeling gene perturbations

Computing attractors on GRNs gives an insight into the cell differentiation process. If the computed Boolean attractors of the GRN have a biological explanation, then the GRN is likely to represent the biological process under investigation. In that case, it would be interesting for biologists to study the results of gene perturbation experiments on the given network.

Gene perturbations (or mutations) can be either in the form of a gene knock-out which leads to constant absence of a protein inside the cell or in the form of a constant high expression of a gene leading to over-production of the corresponding protein. Such mutations may naturally exist in a cell (i.e. inherited from parents) or they could be temporarily induced as a result of a disease or the impact of a drug compound. In both cases, it is interesting for biologists to study the impact of such mutations on the dynamics of the cell. Due to the presence of mutations in a GRN, some sections of the pathways may lose their dynamics completely. Therefore, what may seem as a mutation in a single gene can easily propagate its impact to remotely related genes.

**Example 3.4.1** A synthetic GRN of Figure 3.2(a) in the presence of the knocked-down and the over-expressed node  $A$  is as shown in Figures 3.8(a) and 3.8(b) respectively. The modified pathway can obviously result in a different set of steady states than those in the *wild-type* (i.e. unperturbed) GRN. In Figure 3.8(a), when  $A$  is knocked-down, node  $E$  will always stay at the low expression and node  $C$  will lose its activity towards  $B$ . Hence, the



**Figure 3.8:** Dashed edges does not form a part of the dynamics of the GRN due to mutation in one of the participating node. (a) Modified GRN with the knocked-down node  $A$ . (b) Modified GRN with the over-expressed node  $A$ .

node  $B$  always stay at low expression state. This in turn leads to low expression of  $D$ , which further causes  $C$  to always stay in high expression state. With all the nodes permanently fixed at one expression state, the network loses its ability to produce multiple attractors. A similar explanation can be extended to over-expressed nodes  $A$  in Figure 3.8(b). ■

In the next section, we extend our Boolean model of GRNs to perform *in silico* perturbation experiments. A gene (or protein) in a GRN can exist in one of the following three state:

1. **Over-Expression.** This represents the constant expression of a gene at a high activation level. In Boolean logic, this means that the gene is “ON” or “1” all the time.
2. **Knock-Down.** This represents the case when a gene is silenced and it does not participate in the network dynamics. That means gene is “OFF” or at level “0” all the time.
3. **Wild-Type.** An unperturbed gene (i.e. neither over-expression nor knock-down) is said to be present in the wild-type condition.

The over-expression and knock-down of genes in GRNs, have a similar notion in digital circuits as *stuck-at-1* and *stuck-at-0* faults respectively.

### 3.4.1 Problem formulation

We modify Boolean Equations 3.1,3.2,3.3 and 3.5 to encode knowledge about all possible gene-perturbations in the Boolean model and then, during the analysis phase, we select the genes to be perturbed dynamically. Encoding

this information in the model itself helps in sharing information between different perturbation experiments. Also, such a modeling approach permits the computation of the set of all minimal gene perturbations that can generate a desired steady state. The formulation shown below helps in identifying all such minimal gene perturbation sets without explicitly enumerating and simulating all the possible gene perturbations.

In the presence of perturbations, each node in a GRN can exist in one of the following three states: wild-type, knocked-down and over-expressed. In addition to Boolean vector  $\mathbf{x}^t$ , that was used in previous sections for representing the expression state of all the nodes of a GRN, we use two additional  $N$  bit Boolean vectors  $\mathbf{x}^\downarrow$  and  $\mathbf{x}^\uparrow$  to represent knocked-down and over-expressed nodes respectively in the GRN. If a bit  $i$  of  $\mathbf{x}^\downarrow$  (or  $\mathbf{x}^\uparrow$ ) evaluates to 1 (i.e.  $x_i^\downarrow = 1$  or  $x_i^\uparrow = 1$ ), then it means that the gene  $i$  is knocked-down (or over-expressed). If both  $x_i^\downarrow$  and  $x_i^\uparrow$  evaluate to 0 or 1 for any given  $i$  then the node  $i$  is modeled as a wild-type node. This constraint ensures that a node is not both knocked-down and over-expressed at the same time. To encode this information, we use the modified Boolean variables  $\tilde{x}_i$ 's as in equation 3.9.

$$\tilde{x}_i = \left\{ x_i \wedge \left( \neg x_i^\downarrow \vee x_i^\uparrow \right) \right\} \vee \left( \neg x_i^\downarrow \wedge x_i^\uparrow \right) \quad (3.9)$$

Equation 3.9 states that if the gene  $i$  has been knocked out (i.e. if  $x_i^\downarrow = 1$ ), then  $\tilde{x}_i = 0$ . If the gene  $i$  is over-expressed (i.e. if  $x_i^\uparrow = 1$ ), then  $\tilde{x}_i = 1$ . If the gene is wild-type (i.e. both  $x_i^\downarrow$  and  $x_i^\uparrow$  either evaluate to 0 or 1) then  $\tilde{x}_i = x_i$ . Equation 3.9 accommodates the perturbation information in the input expression of a node. To specify the perturbation information in the Boolean function that defines the expression of a node, we modify Equations 3.1 and 3.2 to Equations 3.10 and 3.12 respectively.

$$\tilde{x}_i(t+1) = \left\{ x_i(t+1) \wedge \left( \neg x_i^\downarrow \vee x_i^\uparrow \right) \right\} \vee \left( \neg x_i^\downarrow \wedge x_i^\uparrow \right) \quad (3.10)$$

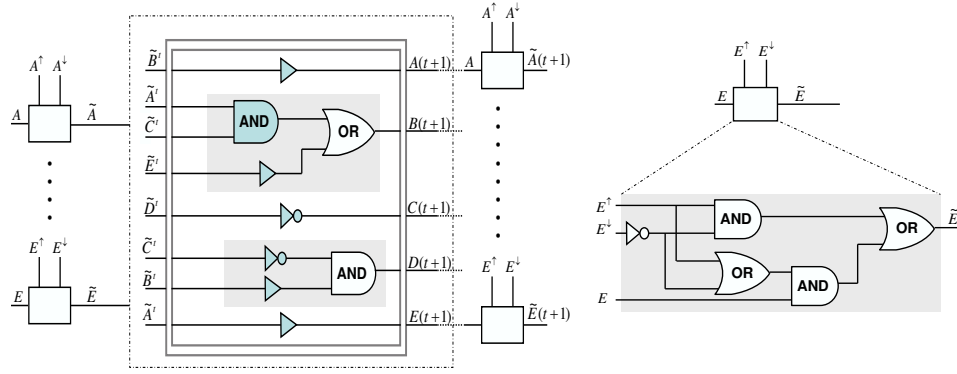
$$x_i(t+1) = \left( \bigvee_{l=1}^n f_{x_{i,l}}^{ac}(t) \right) \wedge \neg \left( \bigvee_{l=1}^n f_{x_{i,l}}^{in}(t) \right) \quad (3.11)$$

$$f_{x_{i,l}}^{ac,in}(t) = \left( \bigwedge_{j=1}^p \tilde{x}_j^{ac,t} \right) \wedge \left( \bigwedge_{j=1}^p \neg \tilde{x}_j^{in,t} \right) \quad (3.12)$$

Equation 3.10 states that if gene  $i$  is over-expressed (i.e. if  $x_i^\uparrow = 1$ ), then  $x_i(t+1) = 1$ . If gene  $i$  has been knocked-down (i.e. if  $x_i^\downarrow = 1$ ), then  $x_i(t+1) = 0$ . If the gene is wild-type (i.e. both  $x_i^\downarrow$  and  $x_i^\uparrow$  either evaluate to 0 or 1), then Equation 3.10 is same as Equation 3.1. Equation 3.12 is the counterpart of Equation 3.2, with the modified variables  $\tilde{x}_i$ .

**Example 3.4.2** The modified Boolean gates mapped synthetic GRN of Figure 3.2(b) is shown in Figure 3.9. As one can see from this figure, extra Boolean logic is appended both at the input and at the output to reflect the





**Figure 3.9:** Modified Boolean gates to accommodate the gene perturbation experiments.

choice between over-expression and knock-down for each input and output variable. ■

Transition functions for the synchronous model given in Equations 3.3 is modified to Equation 3.13 while Equation 3.4 does not changes for the perturbation model.

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \{x_i^{t+1} \leftrightarrow \tilde{x}_i(t+1)\} \quad (3.13)$$

For the asynchronous model, Equation 3.5 is modified to Equation 3.14 while, the rest of the equations for the asynchronous models remain the same as in Equations 3.6-3.8.

$$TP_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \{x_i^{t+1} \leftrightarrow \tilde{x}_i(t+1)\} \wedge \bigwedge_{j \neq i} \{x_j^{t+1} \leftrightarrow \tilde{x}_j\} \quad (3.14)$$

Using the above formulation, one can model multiple gene perturbations in a GRN. A set of perturbations define a single experiment. Multiple experiments spaced over different time points (also referred to as levels of experiments) can also be performed using the above formulation. The modified transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  represents the relation between current state and the next state in the presence of any possible gene perturbation. Given a perturbation experiment (which specifies the set of perturbations to be performed), we restrict the transition function state space to only those perturbations which are part of the experiment and compute attractors on that restricted state space. For this we define three Boolean functions  $f^{\mathbf{x}^\downarrow}$ ,  $f^{\mathbf{x}^\uparrow}$  and  $f^p$  to represent information of the knocked-down, over-expressed and perturbed genes respectively.  $f^p$  is further expressed as a function of  $f^{\mathbf{x}^\downarrow}$  and

$f^{x^\uparrow}$ . These functions are given in Equations 3.15-3.17.

$$f^p = f^{x^\downarrow} \wedge f^{x^\uparrow} \quad (3.15)$$

$$f^{x^\downarrow} = \left( \bigwedge_{i:x_i^\downarrow=1} x_i^\downarrow \right) \wedge \left( \bigwedge_{i:x_i^\downarrow=0} \neg x_i^\downarrow \right) \quad (3.16)$$

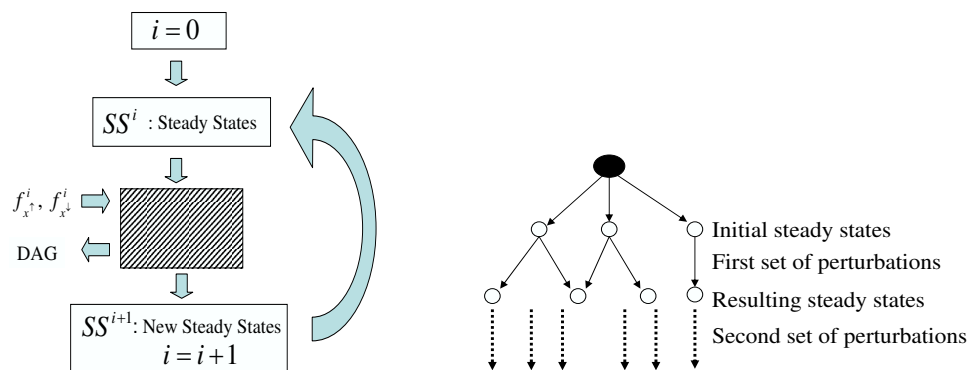
$$f^{x^\uparrow} = \left( \bigwedge_{i:x_i^\uparrow=1} x_i^\uparrow \right) \wedge \left( \bigwedge_{i:x_i^\uparrow=0} \neg x_i^\uparrow \right) \quad (3.17)$$

In the next section, we describe the algorithm to perform perturbation experiments using the above formalism.

### 3.4.2 An algorithm for gene perturbations

A biological experiment often involves more than one perturbations either concurrently or in the time spaced manner. Let us define the set of concurrent perturbations as a single *test* and a sequence of tests as an *experiment*. Tests in an experiment are always performed at different time points defined by their sequence specified in the input experiment file. With this generalised experiment model, the basic idea behind the gene perturbation algorithm can be summarised as in Figures 3.10.

In Figure 3.10(a), black box computes steady states for every test, following the sequence of tests specified in the experiment. The first test in an experiment is always said to be wild-type (i.e. with no perturbations). Starting from the second test, the black box in Figure 3.10(a) checks if the steady states from the previous test can reach the steady states of the current test. The output from this black box is in the form of a DAG (*directed acyclic graph*) as shown in Figure 3.10(b). In Figure 3.10(b), starting from the one level after



(a) Summary of gene perturbation algorithm

(b) Steady state transitions

**Figure 3.10:** (a) Flowchart of gene perturbation algorithm. (b) Directed Acyclic Graph (DAG) showing the transitions among steady states.

---

**Algorithm 5:** Algorithm for *in-silico* gene perturbation experiments
 

---

**Input** : Genetic Network and perturbation experiments.

**Output:** DAG representing steady state analysis.

```

1 begin
2   compute  $T_{sync}$  and  $T_{async}$ 
3   for  $i = 0$  to  $L$  do
4     compute  $f_i^{x^\downarrow}$ ,  $f_i^{x^\uparrow}$  and
5      $f_i^p = f_i^{x^\downarrow} \wedge f_i^{x^\uparrow}$ 
6      $T'_{sync} = T_{sync} \wedge f_i^p$ 
7      $T'_{async} = T_{async} \wedge f_i^p$ 
8      $SS^i[\ ] = all\_attractors(T'_{sync}, T'_{async})$ 
9     if  $i \neq 0$  then
10      for  $k = 0$  to  $SS^{i-1}.size()$  do
11         $FR \leftarrow forward\_set(SS^{i-1}[k], T_{async})$ 
12        for  $j = 0$  to  $SS^i.size()$  do
13          if  $SS^i[j] \subseteq FR$  then
14            Draw an edge between nodes  $SS^{i-1}[k]$  and  $SS^i[j]$ 
15 end

```

---

the root node of the DAG, steady states results are shown in the sequence in which the tests are specified in the experiment. First DAG level gives all the steady states in un-perturbed network, second DAG level gives steady states after first perturbations and so on. Edges among the nodes represents the possible transitions among steady states in the presence of the perturbation. One should note that while steady states can not transition among each other in the absence of a perturbation, they can transition among each other when some nodes are perturbed. This change in stable behavior is captured by the absence of edges between nodes at the same level and presence of edges among the nodes at different levels of DAG in Figure 3.10.

In Figure 3.10, some of the steady states in different levels might be same or may have some measure of similarity. This similarity measure can be computed by counting the percentage of genes which have the same level of expression in two different steady states. Here we compute similarity measure between steady states of first level and steady states of all the other levels. This way one can make more sense out of the perturbation experiments results and can make conclusions such as: the system moves from steady state A to steady state B on perturbing gene X, where A and B are the steady states in the original un-perturbed network.

Figure 3.10 is formally described in Algorithm 5. In this algorithm, the main loop in Lines 3-13 is iterated over the sequence of tests in the experiment. For every test  $i$ , corresponding functions  $f_i^{x^\downarrow}$ ,  $f_i^{x^\uparrow}$  and  $f_i^p$  are constructed using

Equations 3.15-3.17 in Line 4 and 5. Then the transition function is restricted to the state space defined by this perturbation experiment (Lines 6-7). The attractors are computed on the perturbed network in Line 8 using the combined synchronous-asynchronous method. Once the attractors are found, we compare the forward reachability of attractors of the previous test with the attractors of the current level of a perturbation experiment. This is done in Lines 9-14. For every attractor computed in the previous level, i.e. the test  $i - 1$ , we compute the forward reachable states on the new transition function (Line 11). Then we check all the attractors in the current test  $i$  that are contained in this forward reachable set (line 13). Lines 3-14 can be repeated for different experiments on the same network without having to modify the GRN.

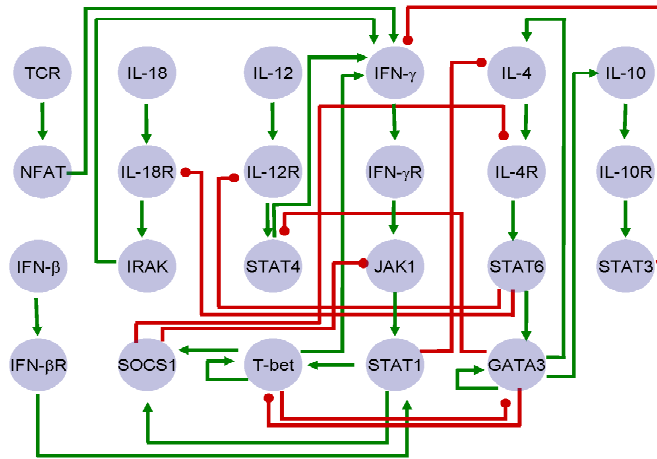
### 3.5 GenYsis toolbox

Algorithms proposed in this thesis are implemented in our modeling toolbox `genYsis`. The software is written in C++ and makes use of the CUDD software package [43] for the BDD manipulation. Executable binaries of `genYsis` for both windows and linux have been made available in the public domain. In the next section, we perform a case study on T-helper GRN showing the application of algorithms developed in this chapter.

### 3.6 Case Study: T Helper network

The vertebrate immune system is made of diverse cell populations; some of them are antigen presenting cells, natural killer cells, B and T lymphocytes. There is a subpopulation of T lymphocytes, the T-helper, or Th, cells that have received much attention from the modeling point of view. Th cells can be divided into precursor Th0 cells and effector Th1 and Th2 cells, depending on the pattern of secreted molecules. Th1 and Th2 cell types play a central role in cellular immunity and humoral responses, respectively. Moreover, immune responses biased towards the Th1 phenotype result in autoimmune diseases, while enhanced Th2 responses originate allergic reactions. At the molecular level, Th1 and Th2 cells can be distinguished by their pattern of cytokine secretion, which are responsible for their central role in cell mediated immunity (Th1 cells) and humoral responses (Th2 cells). Understanding the molecular mechanisms that regulate the differentiation process from Th0 towards either Th1 or Th2 is very important, since an immune response biased towards the Th1 phenotype result in the appearance of autoimmune diseases, and an enhanced Th2 response can originate allergic reactions [23, 94].

There are several factors at the cellular and molecular levels that determine the differentiation of T helper cells. Importantly, the cytokines present in the cellular milieu play a key role in directing Th cell polarization. On



**Figure 3.11:** T-Helper Gene Regulatory Network [102].

the one hand, IFN- $\gamma$ , IL-12, IL-28 and IL-27 are the major cytokines that promote Th1 development [144]. And on the other hand, IL-4 is the major cytokine responsible for driving Th2 responses. Besides the positive role of cytokines in the differentiation process, there exist also a mutual inhibitory mechanism. Specifically, IFN- $\gamma$  play a role in inhibiting the development of Th2 cells, whereas IL-4 inhibits the appearance of Th1 cells. This interplay of positive and negative signals, at both the cellular and molecular levels, creates a complexity that is very suitable for analysis by the modeling approach.

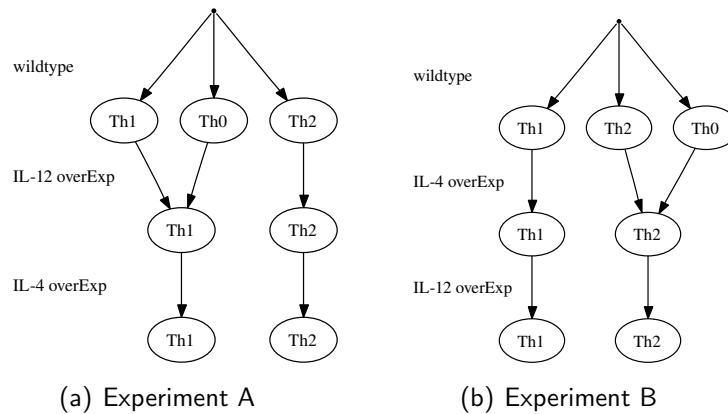
Due to its physiological relevance, there are various mathematical models that have been proposed for describing the differentiation, activation and proliferation of T helper lymphocytes. Most of these models, however, focus on interactions established among the diverse cell populations that somehow modify the differentiation of Th cells [19, 9]. Also, other modeling efforts have been aimed at understanding the mechanism of the generation of antibody and T-cell receptor diversity, as well as the molecular networks of cytokine or immunoglobulin interactions [50, 48]. Recently, there have been some publications on the regulatory network that controls the differentiation of Th cells [102, 99]. The regulatory network presented (reproduced in Figure 3.11) constitutes the most extensive attempt to model the regulatory network controlling the differentiation of Th lymphocytes to date. The topology of the network was derived from published experimental data. The network (Figure 3.11) is made of 23 nodes, 26 positive and 8 negative interactions. Importantly, the model does not need to be seen as metabolic pathway, or a reaction network, but rather as an information processing network.

### 3.6.1 Simulation results

On applying the algorithms developed in this chapter on the T helper cell network of Figure 3.11, three wild type steady states as listed in Table 3.3

**Table 3.3:** Steady states of the T-helper Cell.

Perturbed Genes	Active genes in steady states	Cell Type
wild type	All the genes are inactive	Th0
	IFN- $\gamma$ Tbet SOCS-1 IFN- $\gamma$ R	Th1
	IL-10 IL-10R GATA-3 STAT3 STAT6 IL-4 IL-4R	Th2
IL-12 <sup>+/+</sup>	IFN- $\gamma$ Tbet SOCS-1 IFN- $\gamma$ R IL-12 IL-12R STAT4	Th1
	IL-10 IL-10R GATA-3 STAT3 IL-12 STAT6 IL-4 IL-4R	Th2
IL-4 <sup>+/+</sup>	IFN- $\gamma$ Tbet SOCS-1 IFN- $\gamma$ R IL-4	Th1
	IL-10 IL-10R GATA-3 STAT3 STAT6 IL-4 IL-4R	Th2

**Figure 3.12:** Results of Gene Perturbation Experiments

are found. These steady states correspond to the molecular profiles observed in Th0, Th1 and Th2 cells respectively. The first steady state reflects the pattern of Th0 cells, which are precursor cells that do not produce any of the cytokines included in the model (i.e. IFN- $\beta$ , IFN- $\gamma$ , IL-10, IL-12, IL-18 and IL-4). The second steady state represents Th1 cells with high activation of IFN- $\gamma$ , IFN- $\gamma$ R, T-bet and SOCS1. Finally the third steady state corresponds to the activation observed in Th2 cells, with high level of activation of GATA-3, IL-10, IL-10R, IL-4, IL-4R, STAT3 and STAT6. These results also match those published in [102].

Next, the response of Th cells was studied to two consecutive stimuli, first a constant saturating concentration of IL-12, and then changing it to a saturating concentration of IL-4. As shown in Figure 3.12(a), this combination of signals has the result of eliminating the Th0 steady state. If the system is in the Th0 state, the constant activation of IL-12 moves it to the Th1 state, where it stays even after the inactivation of IL-12 and the constant presence of IL-4. By contrast, if the system starts in the Th1 or Th2 states, the two consecutive signals are incapable of moving the system to another attractor. The steady state profile on IL-12 and IL-4 over-expression are shown in Table 3.3. Figure 3.12(b), shows the simulation results on applying the same perturbations as described above in the reverse order (i.e. activating IL-4 to its

highest level, and then inactivating it and activating IL-12 instead). Results are the same as before, shown in Figure 3.12(b). The only difference is that in this simulation, the network in the Th0 states receives the IL-4 and moves to Th2, where it stays after the elimination of IL-4 and the activation of IL-12.

These simulations show that Th0 state is unstable under the perturbation of IL-12 or IL-4, which act as differentiation signals to take the system to the Th1 or Th2 states, respectively. By contrast, the Th1 and Th2 states are stable under the perturbation of the IL-4 and IL-12 nodes. These results are in total agreement with experimental data [94] and reported simulations of the Th network using a different mathematical framework [102]. In the literature, modeling of Th cell differentiation at the molecular level has been shown to be very useful to bring insight into the origin of the unexpected phenotypes. By using the algorithms proposed in this chapter one can easily perform such simulations *in silico*.

## 3.7 Summary

In this chapter, we started with an informal description of biological functionalities in a GRN in terms of Boolean functions. We then introduced a modeling approach for GRNs based on Boolean algebra and discrete methods, reminiscent of the style used in digital circuit synthesis and verification. A common formalism for both synchronous and asynchronous dynamics modeling approach was then introduced. Differences between the two dynamic models and their impact on computational results were discussed. It was shown that asynchronous modeling can be a far more computation intensive than the synchronous modeling of the same GRN. An algorithm based on combined synchronous-asynchronous modeling was then introduced to reduce the computation time for attractors in asynchronous circuits. Finally, the Boolean formulation was modified to simulate gene perturbation experiments *in silico*. By showing a biologically motivated example of T-Helper GRN, we demonstrated a practical application of algorithms proposed in this chapter

The formalism proposed in this chapter assumes that genes (or proteins) can only exist in two expression states (i.e. active and inactive). This assumption is not biologically impractical as most of the protein signaling in practice have a switch-like behavior. However, one can easily come up with a few examples of proteins that show different functionalities at more than two levels of activation. To address this issue, the next chapter extends the modeling approach proposed in this chapter to multiple activations levels of nodes in a GRN.





---

# Multiple Valued Modeling

---

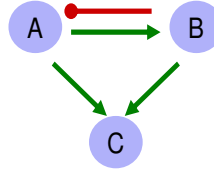
# 4

Boolean modeling of GRNs inherently assumes the binary nature of cellular signaling represented by the protein being active or inactive. Although the assumption of only two states of protein activity is not entirely inappropriate in biology, at least a few proteins are known to biologists that demonstrate different functionalities when present in more than two activity states. For example, activation of a pro-survival protein *Akt* occurs via its phosphorylation of two residues *Ser473* and *Thr308*. Mutational analysis has shown that while phosphorylation at *Thr308* is sufficient to activate *Akt*, phosphorylation at both its residues is required for the maximal activation of *Akt* [32, 121]. To model such a behavior in GRNs, it is important that *Akt* is modeled at more than two levels of activity in a GRN.

This chapter extends the formalism proposed for Boolean modeling in the previous chapter so as to model GRNs where genes (or proteins) can have three or more activation states. The chapter starts with an introduction to *1-hot encoding*, a commonly used methodology for mapping multiple valued models on Boolean models. Sections 4.2 and 4.3 show the application of multiple valued models on the T-Helper cells and *Arabidopsis thaliana* GRNs. Finally, Sections 4.4 and 4.5 extend the multiple valued formulation to sigmoid function representation of protein activation functions.

## 4.1 1-hot encoding of GRNs

Multiple valued modeling of GRNs where a gene (or protein) can have more than two levels of expressions can also be modeled using the algorithms given in Chapter 3. Unlike in Boolean models, where a single function can specify rules to activate a node at both low and high expression states, a different function is required to specify the activation of nodes at each activity levels in multi-



**Figure 4.1:** A synthetic GRN.

**Table 4.1:** Multiple valued transition rules for the GRN in Figure 4.1.

Gene Name	Transition rules		
	Low	Medium	High
c	a = l, b = l	a = m, b = m	a = m, b = h
c	a = l, b = m	a = l, b = h	a = h, b = m
c	a = m, b = l	a = h, b = l	a = h, b = h
a	b = h	b = m	b = l
b	a = l	a = m	a = h

valued logic. For example Table 4.1 specifies the input output relationship for the activation of the node  $C$  in the GRN of Figure 4.1.

Nodes that have multi-valued activation states (such as low, medium and high) can be encoded into Boolean representation using the concept of 1-hot encoding. Under 1-hot encoding, each variable that can take up to  $n$  values is represented by a Boolean vector of length  $n$ . Therefore, unlike Boolean logic where a gene expression state is represented by a Boolean variable  $x_i$ , in the multiple valued logic, a gene expression is represented by using a binary vector  $\mathbf{x}_i$  of length  $n$ . The  $j^{\text{th}}$  entry of the vector  $x_i$  is 1 (i.e.  $x_{i,j} = 1$ ) if the variable  $i$  takes the expression level  $j$ . Otherwise  $x_{i,j} = 0$ . For example, if a gene  $i$  has three expression levels : low, medium and high (given by 0, 1 and 2), then these levels can be represented as 001, 010 and 100 respectively. This encoding scheme is commonly known as *1-hot encoding* [46].

In the multiple valued modeling of GRNs, in addition to the connectivity graph (Figure 4.1), a rule table in the format of Table 4.1 is given. If we use 1-hot encoding to implicitly represent each column of Table 4.1, we can treat each expression state  $i$  of the node  $c$  as a Boolean function  $x_{c,i}(t+1)$  of the expression states of the input nodes  $a$  and  $b$ . For example Equation 4.1 gives the Boolean function for the medium expression state of the node  $c$ .

$$x_{c,1}(t+1) = (x_{a,1} \wedge x_{b,1}) \vee (x_{a,0} \wedge x_{b,1}) \vee (x_{a,1} \wedge x_{b,0}) \quad (4.1)$$

Equation 4.1 can be generalized to Equation 4.2 and 4.3, where  $x_{i,j}^t$  is the boolean variable for the  $j^{\text{th}}$  expression level of the gene  $i$  at the time instant

$t$ . Equations 4.2 and 4.3 are the counterpart of Equations 3.1 and 3.2 in Section 3.2 for Boolean modeling of GRNs.

$$x_{i,j}(t+1) = \left( \bigvee_{p=1}^{n_{i,j}} f_{i,j,p}(t) \right) \quad (4.2)$$

$$f_{i,j,p}(t) = \left( \bigwedge_{k \in R_{i,j,p}} x_k^t \right) \quad (4.3)$$

$$x \in \{0, 1\}$$

$\wedge$  and  $\vee$  represent logical AND and OR

$n_{i,j}$  is the number of rules for the  $j^{\text{th}}$  logic level of gene  $i$

$R_{i,j,p}$  is the tuple  $\{a, b\}$  of variables and logic levels in transition rules

Similar to Boolean modeling, a state of the network is represented by a Boolean vector  $\mathbf{x}^t$  of size  $\sum_{i=1}^N l_i$ , where  $N$  is the number of genes in the network and  $l_i$  is the number of expression levels of the gene  $i$ . Each gene has  $l_i$  continuous bits in the vector  $\mathbf{x}^t$  representing the corresponding logic levels. Only one of these  $l_i$  bits can be 1 since a gene can only be at one expression level at any given instant of time. Another similar Boolean vector  $\mathbf{x}^{t+1}$ , is used to represent the status of the genes at the next time step. The transition function  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  given in Chapter 3 for the synchronous model (Equations 3.3) and the asynchronous model (Equations 3.7) are slightly modified to Equations 4.4 and Equations 4.5 respectively. The combined transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  remains the same as in Equations 3.4 and 3.8 for the synchronous and asynchronous models respectively.

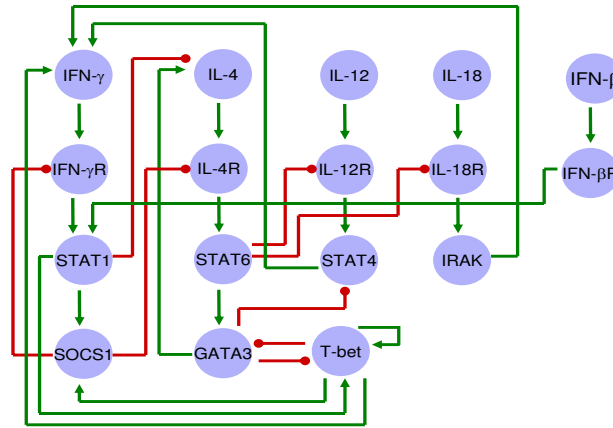
$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \bigwedge_{j=1}^{l_i} (x_{i,j}^{t+1} \leftrightarrow x_{i,j}(t+1)) \quad (4.4)$$

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \left\{ \bigwedge_{j=1}^{l_i} (x_{i,j}^{t+1} \leftrightarrow x_{i,j}(t+1)) \right\} \wedge \bigwedge_{k \neq i} \left\{ \bigwedge_{j=1}^{l_k} (x_{k,j}^{t+1} \leftrightarrow x_{k,j}^t) \right\} \quad (4.5)$$

With the modified transition functions, algorithms as given in Sections 3.2.4 and 3.3.1 for Boolean models can be used for synchronous, asynchronous and combined synchronous-asynchronous modeling of the multi-valued GRNs. In the next two sections, we model multi-valued T-Helper and Arabidopsis thaliana GRNs using the above formalism.

## 4.2 Multiple valued T Helper network

We modeled the multi-valued T-helper Network (Figure 4.2) introduced in [99]. This network has four genes at three levels of activation: low, medium and high. All the other genes have only two levels: low and high. The transition



**Figure 4.2:** T-Helper Gene Regulatory Network.[99]

**Table 4.2:** Steady States Observed in Wild Type and Mutated T Helper Gene Network (Figure 4.2).

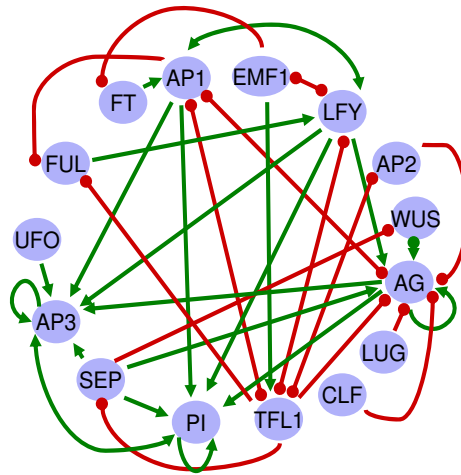
Perturbed gene	Active genes in steady states	Cell Type
Wild Type	All the genes at low activation level	Th0
	IFN- $\gamma$ (h) T-bet(h) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	IFN- $\gamma$ (m) T-bet(m) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	GATA-3(h) STAT-6(h) IL-4R(h) IL-4(h)	Th2
IL-12 <sup>+/+</sup>	IFN- $\gamma$ (h) T-bet(h) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	IL-12(h) IL-12R(h) STAT-4(h)	
	IFN- $\gamma$ (m) T-bet(m) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	IL-12(h) IL-12R(h) STAT-4(h)	
	GATA-3(h) STAT-6(h) IL-4R(h) IL-4(h) IL-12(h)	Th2
IL-4 <sup>+/+</sup>	IFN- $\gamma$ (h) T-bet(h) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	IL-4(h)	
	IFN- $\gamma$ (m) T-bet(m) SOCS-1(h) IFN- $\gamma$ R(m) STAT-1(m)	Th1
	IL-4(h)	
	GATA-3(h) STAT-6(h) IL-4R(h) IL-4(h) IL-12(h)	Th2

rules describing the expression of different genes are given in Table 4.3. When this network was run through *genYsis*, it found four wild type (without any mutation) steady states shown in Table 4.2. These steady states match the ones reported in [99].

We also tried two mutations, IL-12<sup>+/+</sup> (i.e. IL-12 over-expressed) and IL-4<sup>+/+</sup> (i.e. IL-4 over-expressed). The steady states are listed in Table 4.2. Both the mutations remove the Th0 cell state from the GRN. This is also in correspondence with the results reported in [99] and have similarities with Boolean steady states reported in Table 3.3 in Chapter 3.

Table 4.3: Multiple valued transition rules for T-helper network.

Name	Transition rules		
	Low	Medium	High
IFNg	IRAK(h) default	STAT4(h) STAT4(h),T-bet(m) IRAK(h),T-bet(m) T-bet(m)	STAT4(h),IRAK(h) STAT4(h),IRAK(h),T-bet(m) STAT4(h),IRAK(h),T-bet(h) STAT4(h),T-bet(h) IRAK(h),T-bet(h) T-bet(h)
IRAK	default		IL18R(h)
STAT4	IL12R(h),GATA3(h) GATA3(h) default		IL12R(h)
IL4	STAT1(m) STAT1(m),GATA3(h) STAT1(h) STAT1(h),GATA3(h) default		GATA3(h)
STAT1	default	IFNbR(h) IFNgR(m) IFNbR(h),IFNgR(m)	IFNgR(h) IFNbR(h),IFNgR(h)
GATA3	STAT6(h),T-bet(m) STAT6(h),T-bet(h) T-bet(m) T-bet(h) default		STAT6(h)
IFNgR	SOCS1(h) default	IFNg(m) IFNg(m),SOCS1(h) IFNg(h),SOCS1(h)	IFNg(h)
SOCS1	default		STAT1(m) STAT1(m),T-bet(m) STAT1(m),T-bet(h) STAT1(h) STAT1(h),T-bet(m) STAT1(h),T-bet(h) T-bet(m) T-bet(h)
IL12	default		
STAT6	default		IL4R(h)
T-bet	STAT1(m),GATA3(h) STAT1(h),GATA3(h) STAT1(h),GATA3(h),T-bet(m) STAT1(h),GATA3(h),T-bet(h) GATA3(h) GATA3(h),T-bet(m) GATA3(h),T-bet(h) default	STAT1(m) STAT1(m),GATA3(h),T-bet(m) STAT1(m),T-bet(m) T-bet(m)	STAT1(m),GATA3(h),T-bet(h) STAT1(m),T-bet(h) STAT1(h) STAT1(h),T-bet(m) STAT1(h),T-bet(h) T-bet(h)
IL4R	IL4(h),SOCS1(h) SOCS1(h) default		IL4(h)
IL12R	IL12(h),STAT6(h) STAT6(h) default		IL12(h)
IL18	default		
IFNbR	default		IFNb(h)
IL18R	IL18(h),STAT6(h) STAT6(h) default		IL18(h)
IFNb	default		



**Figure 4.3:** Arabidopsis thaliana Gene Regulatory Network. [20]

The analysis performed on the T-Helper model permits the identification of all the stable states observed in the biological system, specifically under wild-type conditions. It is straightforward to modify the model so as to describe the situation where there are knock-downs and over-expressions. This capacity to simulate mutants helps both to validate the model and to help interpret the behavior in the presence of mutations and drug stimuli.

Published quantitative data on the expression of the molecules represented in the Th regulatory network is currently lacking. Hence, it would be very instructive to model the Th network at different levels of granularity with respect to the levels of activation of its nodes, so as to know which steady states are obtained regardless of the underlying modeling approach. Moreover, it is important to compare the result of mutants in the model, so as to validate it against experimental data. GenYsis provides a way to biologists to perform these experiments *in silico* and test the validity of the network with respect to the experimental data.

### 4.3 Arabidopsis thaliana network

Flowers of *Arabidopsis thaliana* are formed by four concentric whorls of organs made of four sepals (the outermost whorl), four petals, six stamens and two fused carpels (the innermost whorl). This organization of the flower can be disrupted by mutations in a series of genes. The analysis of such mutations led to the proposition of a combinatorial schema, called the “ABC model”, which has been used extensively to describe the morphology of Arabidopsis flowers, both in a wild type and mutant backgrounds [35]. The ABC model postulates the existence of three different abstract activities, namely A, B, and C, each of which is present in sets of two adjacent whorls. Whichever the nature of the molecular mechanism, the particular combination of these activities determines

the identity of the organs that will develop in a particular whorl. Specifically, the sole presence of the A activity will determine the differentiation of the underlying tissue into sepals. The combination of A and B activities, however, determine the differentiation of petals. The combination of B and C leads to a production of stamens. And finally, the C activity by itself determines the development of carpels. Additionally, the ABC model postulates a mutual inhibition between activities A and C, such that when function A is absent function C substitutes it and vice versa.

Many genes involved in *Arabidopsis thaliana* flowering and flower morphogenesis have been identified. This has permitted the cloning and analysis of expression patterns of the genes. Moreover, there is already a wealth of information related to the phenotype associated with alteration in gene expression. All the known experimental information lead to the proposition of the regulatory network that controls the flower morphogenesis in Arabidopsis [101, 100]. This first model was later enlarged by the incorporation of new genes and interactions [20]. The initial versions of the flowering model used binary variables to represent the activation of genes. The more recent version of the model [20] used both two- and three-valued variables to represent the levels of gene activation. In this case, the model showed patterns of expression that could be compared directly with those observed not only in the mature flower, but also in the inflorescence meristems and floral organ primordia. Finally, in all versions of the Arabidopsis model it was possible to simulate the effect of null mutations, obtaining results that were qualitatively correct with the published experimental data.

Here, we show the application of **genYsis** on the arabidopsis network published in [20] with both two and three levels of expression of gene activation. The network used is as shown in Figure 4.3. The rules encoding the interactions of genes are in a similar format as for the T Helper Cell Network in the previous section.

GenYsis found 10 wild type steady states on the GRN in Figure 4.3. These steady states are listed in Table 4.4 and match the ones reported in [20]. Of the 10 steady states listed in Table 4.4, four steady states represent the inflorescence meristems and the remaining six steady states correspond to the formation of different parts of the flower, namely: *stamens* (St), *carpel* (Car), *petals* (Pt) and *sepals* (Sp). In addition to the wild type experiment, we tried AP3 and AP2 gene knock-downs. The steady states reported are listed in Table 4.4. As one can see from Table 4.4 that by knocking down AP3, GRN does not show the steady states corresponding to formation of stamens and petals. On the other hand, on knocking down AP2, steady states corresponding to sepals and petals are missing. This biologically corresponds to the loss of ability of the formation of these organs in the flower carrying the gene mutations. These steady states match the cell states reported in experimental data published [156, 115] and [86]. The same results were also reported in [20].

Table 4.4: Steady States Observed in Wild Type and Mutated *Arabidopsis thaliana* GRN.

Mutation	Gene Name																Cell Type
	FT	EMF1	LFY	TFL1	API	FUL	AP2	SEP	AG	PI	AP3	WUS	LUG	CLF	UFO		
Wild Type	0	1	0	2	0	0	0	0	0	0	0	1	1	1	1	1	<i>Infl3</i>
	1	0	2	0	0	2	1	1	2	2	2	0	1	1	1	1	<i>St1</i>
	1	0	2	0	0	2	1	1	2	1	0	0	1	1	1	0	<i>Car</i>
	1	0	2	0	2	0	1	1	0	2	2	0	1	1	1	1	<i>Pe1</i>
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	1	0	<i>Infl4</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	1	<i>Infl2</i>
	1	0	2	0	2	0	1	1	0	0	0	0	1	1	1	0	<i>Sep</i>
	1	0	2	0	0	2	1	1	2	2	2	0	1	1	1	0	<i>Pe2</i>
	1	0	2	0	2	0	1	1	0	2	2	0	1	1	1	0	<i>St2</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	0	<i>Infl1</i>
	<i>AP3</i> <sup>-</sup>	0	1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl4</i>
1		0	2	0	2	0	1	1	0	0	0	0	1	1	0	<i>Sep</i>	
1		0	2	0	0	2	1	1	2	1	0	0	1	1	0	<i>Car</i>	
1		0	2	0	2	0	1	1	0	0	0	0	1	1	1	<i>Sep</i>	
1		0	2	0	0	2	1	1	2	1	0	0	1	1	1	<i>Car</i>	
0		1	0	2	0	0	0	0	0	0	0	1	1	1	1	<i>Infl3</i>	
0		1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl1</i>	
0		1	0	2	0	0	0	0	0	0	0	0	1	1	1	<i>Infl2</i>	
0		1	0	2	0	0	0	0	0	0	0	0	1	1	1	1	<i>Infl2</i>
1		0	2	0	0	2	0	1	2	2	2	0	1	1	1	1	<i>St1</i>
0		1	0	2	0	0	0	0	0	0	0	1	1	1	1	1	<i>Infl3</i>
<i>AP2</i> <sup>-</sup>	1	0	2	0	0	2	0	1	2	1	0	0	1	1	0	<i>Car</i>	
	0	1	0	2	0	0	0	0	0	0	0	1	1	1	0	<i>Infl4</i>	
	1	0	2	0	0	2	0	1	2	1	0	0	1	1	1	0	<i>St2</i>
	1	0	2	0	0	2	0	1	2	2	2	0	1	1	1	0	<i>Infl1</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	0	<i>Infl1</i>
	0	1	0	2	0	0	0	0	0	0	0	0	1	1	1	0	<i>Infl1</i>



## 4.4 Sigmoid function

In the previous sections of this chapter, there was an inherent assumption that the multi-valued rules such as the ones given in Table 4.3 can be easily abstracted from the gene-gene or gene-protein interactions knowledge available in the literature. While information such as activation or inhibition of gene (or protein) by other proteins is readily available, extracting such multi-valued rules from this information may not be very straightforward. In such a situation, one has the knowledge about the connectivity of the network but a little knowledge about the underlying multiple valued rules. However, we know that kinetic laws such as Hill function, Michaelis-Menten and sigmoid functions are often used for specifying the continuous dynamics of the expression of genes (or proteins) in a GRN. By discretising the functions defined by these kinetic laws at three or more expression levels, one can automatically extract multiple valued rules. In this section, we extend the Boolean formalism of GRNs, proposed in Chapter 3 to one such kinetic function, the sigmoid function (Equation 4.6) introduced in [99], to represent the relationship between the expression of input genes and the output gene.

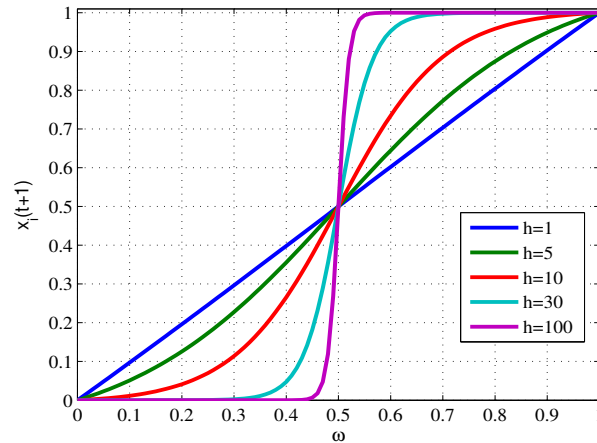
$$x_j(t+1) = \frac{-e^{0.5h} + e^{-h(\omega_j-0.5)}}{(1 - e^{0.5h})(1 + e^{-h(\omega_j-0.5)})} \quad (4.6)$$

$$\omega_j = \begin{cases} \left[ 1 - k_j^{in} \left( \frac{1+\sum \beta_n}{\sum \beta_n} \right) \left( \frac{\sum \beta_n x_n^{in}}{1+\sum \beta_n x_n^{in}} \right) \right] \times \left[ k_j^{ac} \left( \frac{1+\sum \alpha_n}{\sum \alpha_n} \right) \left( \frac{\sum \alpha_n x_n^{ac}}{1+\sum \alpha_n x_n^{ac}} \right) \right] \\ \left[ 1 - \left( \frac{1+\sum \beta_n}{\sum \beta_n} \right) \left( \frac{\sum \beta_n x_n^{in}}{1+\sum \beta_n x_n^{in}} \right) \right] \\ \left[ \left( \frac{1+\sum \alpha_n}{\sum \alpha_n} \right) \left( \frac{\sum \alpha_n x_n^{ac}}{1+\sum \alpha_n x_n^{ac}} \right) \right] \end{cases} \quad (4.7)$$

$$0 \leq \alpha, \beta, k, x \leq 1$$

Equation 4.6 has the value of expression of a gene between '0' and '1'. The first expression for  $\omega$  is used if both activators and inhibitors are acting on a gene, second and third expressions are used if only inhibitors and only activators respectively are acting on a gene. In the former case, the gene may have some weight associated to inhibiting effect and activating effect, given by  $k_j^{in}$  and  $k_j^{ac}$  respectively. Parameters  $\alpha$  and  $\beta$  are the weights on edges in the GRN. Parameter  $h$  is called the *gain* of the function and decides the steepness of the curve.

Figure 4.4 demonstrates the sigmoid behavior of the expression of  $x_i(t+1)$  in Equation 4.6 with different values of the parameter  $h$ . As one can see from Figure 4.4, sigmoid function approximates to a Boolean function if the parameter  $h$  in Equation 4.6 is chosen appropriately. Since, the sigmoid function in Equation 4.6 is a closest non-linear continuous function to the Boolean function, it serves as a right model to infer multiple-valued rules for specifying the dynamics of the expression of a node in a GRN.



**Figure 4.4:** Simulation of Equation 4.6, displaying the sigmoid approximation of discrete Boolean function with varying values of the gain parameter  $h$ . [99]

**Table 4.5:** Steady states of the T-helper GRN in Figure 3.11.

Active genes in steady states							Cell Type
All the genes are inactive							Th0
IFN- $\gamma$ (m)	T-bet (h)	SOCS-1 (h)	IFN- $\gamma$ R (m)				Th1
IFN- $\gamma$ (m)	T-bet (m)	SOCS-1 (m)	IFN- $\gamma$ R (m)				Th1
IL-10 (h)	IL-10R (h)	GATA-3 (h)	STAT3 (h)	STAT6 (h)	IL-4 (h)	IL-4R (h)	Th2

If all the gene expressions  $x_i$  are to be modeled at  $l_i$  discrete expression levels, then the Equations 4.6 and 4.7 can be discretized at these discrete levels to give the multiple valued rules (in the format used in Section 4.2). We now model again the small T-Helper network in Section 4.2 and the bigger T-helper network used in Section 3.6 by inferring the multiple valued rules using the above sigmoid function. Each node in the GRN is modeled at three levels of activation, specified by  $x_i = 0$ ,  $x_i = 0.5$  and  $x_i = 1.0$  in Equation 4.6. These activation levels qualitatively correspond to the low, medium and high activation levels of a gene (or protein). In the following simulations, the parameter  $h$  is chosen as 3 and all the  $\alpha$  and  $\beta$  are assigned the value 1.

On modeling the T-helper network in Figure 3.11, we find four steady states given in Table 4.5. Two steady states corresponds to Th1 cell state profile and the remaining two steady states represent the Th0 and Th2 cell states. As one can see the the gene expression profiles of Th0, Th1 and Th2 in Table 4.5 resembles the steady state profiles obtained from Boolean models (as given in Table 3.3 in Chapter 3).

Similarly when the automatic extraction of rules is applied on the smaller T-Helper network, which was modeled using biologically inferred multiple-valued rules in Section 4.2, it gives five steady states (Table 4.6). Two steady states represents the Th2 cell state and the remaining two represent the Th0

**Table 4.6:** Steady states of the small T-helper GRN in Figure 4.2.

Active genes in steady states					Cell Type
All the genes at low activation level					Th0
IFN- $\gamma$ (m)	T-bet (h)	SOCS-1 (h)	IFN- $\gamma$ R (m)	STAT-1 (m)	Th1
GATA-3 (h)	STAT-6 (h)	IL-4R (h)	IL-4 (h)		Th2
GATA-3 (m)	STAT-6 (m)	IL-4R (m)	IL-4 (m)		Th2

and Th2 cell states. The Th0, Th1 and Th2 activation profile of steady states in Table 4.6 can be compared with the activation profile of steady states in Table 4.2 obtained when the multi-valued rules were explicitly given.

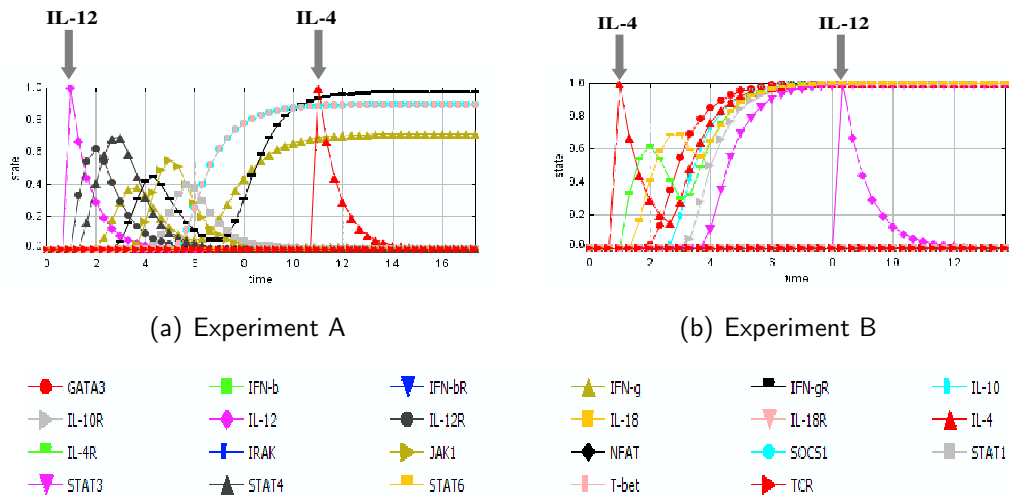
From the above simulations, it is evident that automatic extraction of rules using Equation 4.6, can serve as an alternative for generating multiple values rules when explicit information linking the expression states of genes at more than two activation levels is not available. Another advantage of computing gene expression evolution using the Equations 4.6 and 4.7 is that the expression is continuous between ‘0’ and ‘1’ level and the sigmoid function approximates to Boolean function. Therefore, one can also use a combined Boolean and ODE (represented by the above sigmoid function) technique as we will show in the next section.

## 4.5 Combined Boolean-ODE formalism

The sigmoid function in Equation 4.6 was originally proposed in [99] for combined Boolean-ODE simulations. Dynamics of the network using the sigmoid function in Equation 4.6 can be represented by a system of coupled ODE’s specified in Equation 4.8, where  $\gamma_i$  represents the decay parameter of the expression of a node.

$$\frac{dx_i}{dt} = x_i(t+1) - \gamma_i x_i \quad (4.8)$$

Equation 4.8 is a nonlinear differential equation and can be solved using numerical integration techniques. To compute the steady state behaviour of a GRN, Equation 4.8 is simulated starting from an initial assignment to  $x_i$ ’s, until the system of ODE’s converge to a steady state. However, identifying starting configurations of variable  $x_i$ ’s so as to discover different steady states of the GRN can be a cumbersome task. To address this issue, if the parameter  $h$  in Equation 4.6 is chosen such that the sigmoid function is not very far from the discrete Boolean function then, one can use the steady states identified in Boolean modeling of GRNs as a starting configuration of  $x_i$ ’s in Equation 4.8. This modeling technique has been applied in the simulation toolbox **SQUAD** (*Standardized Qualitative Dynamical Modeling Suite*) which integrates the Boolean and ODE formulation of GRNs. **SQUAD** implicitly makes use of algorithms proposed in this thesis for computing the Boolean steady states and



**Figure 4.5:** Results of Gene Perturbation Experiments

then perform dynamic simulations using Equation 4.8 with Boolean steady states as a starting configuration.

Using the continuous system of differential equations provides additional functionalities on top of Boolean formalism, as the dynamics of GRNs can be visualized in continuous time domain and gene perturbations can be applied at different time points (and for varying time durations) mimicking the behaviour of drug dosage. For example, Figure 4.5 presents the output from SQUAD for the same gene perturbation experiments that were performed in Section 3.6 on Boolean formalism of GRNs. These simulations show that Th0 state is unstable under the perturbation of IL-12 (Figure 4.5(a)) or IL-4 (Figure 4.5(b)), which act as differentiation signals to take the system into the Th1 or Th2 state respectively. By contrast, the Th1 and Th2 states are stable when a second pulse of IL-4 and IL-12 is given (Figures 4.5(a) and 4.5(b) respectively).

The `genYsis` toolbox has been integrated in SQUAD, which also acts as a graphical user interface for algorithms proposed in Chapter 3 for Boolean modeling of GRNs.

## 4.6 Summary

In this chapter, we presented methodologies for modeling multiple-valued GRNs. We started with an extension to the formalism proposed in the previous chapter by encoding multiple levels of gene expressions into Boolean variables using the 1-hot encoding scheme. The modified formalism can make use of algorithms proposed for Boolean modeling for computing the attractors in multi-valued GRNs. Algorithms were applied on the multiple-valued T-Helper GRN for modeling cellular differentiation and on the *Arabidopsis thaliana* GRN for studying the formation of different organs of the flower. Then a sigmoid

---

function based methodology was introduced for extracting the multi-valued transition rules from the Boolean functions describing the gene regulation mechanisms. By avoiding an explicit enumeration of rules that define gene regulation at each activation level, one can apply algorithms developed for Boolean modeling at various levels of discretisation. Finally, algorithms for computing attractors in Boolean modeling were integrated with the sigmoid function based ODEs demonstrating a combined Boolean-ODE modeling approach. The integration of Boolean algorithms of **genYsis** with the ODE formulation in **SQUAD** provides a notion of time scale to the dynamics of GRNs and also facilitates a graphical user interface for **genYsis**.



---

# Probabilistic Boolean Networks

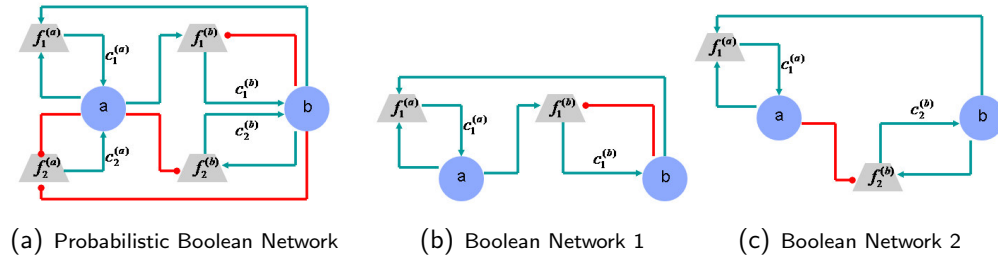
---

# 5

Even though Boolean networks have been used successfully in the past to model various real gene regulatory networks [20, 102, 74], their inherent deterministic nature and Boolean logic have been a central issue of criticism. Non-determinism in gene regulatory networks may exist for various reasons. For instance, a protein may not bind to its operation site (leading to loss of functionality represented by the GRN) or multiple functions may exist to explain the behaviour of a gene in similar circumstances. Such non-deterministic behaviour is difficult to accommodate in Boolean networks. This prompted researchers in [65] to propose a probabilistic extension of Boolean networks termed as *probabilistic Boolean networks* (PBNs).

In a PBN, behaviour of a gene can be described with multiple Boolean functions. Each function has a probability associated with it and there is at least one function corresponding to each gene that can predict its expression as a function of the expressions of the input genes. If all the genes have only one function then a PBN is similar to a BN. Alternatively, a PBN can be seen as a set of BNs. In that case, each BN has a probability equal to the product of the probabilities associated with the Boolean functions of which it is composed. Although most analysis on PBNs are done by looking at the latter description of a PBN, in this chapter we look at the equivalent former description and propose a more suitable mathematical model for efficient analysis of probabilistic gene regulatory networks.

This chapter extends the deterministic Boolean formalism proposed in Chapter 3 for implicit modeling of GRNs to PBNs. The chapter starts with the PBN formulation as proposed by [65] in Section 5.1. Then an improved formalism of PBN is proposed in Section 5.2 which is suitable for implicit representation and traversal of PBNs. Algorithms based on implicit representation are given in Section 5.3 for modeling some interesting functionalities of PBNs.



**Figure 5.1:** (a) A sample PBN. There are two variables  $a$  and  $b$  and four functions  $f_1^{(a)} = a \wedge b$ ,  $f_2^{(a)} = \neg a \wedge \neg b$ ,  $f_1^{(b)} = a \wedge \neg b$  and  $f_2^{(b)} = \neg a \wedge b$ . Arrow headed edges represent activation and circle headed edges represent inhibition. (b) A Boolean network formed by selecting only  $f_1^{(a)}$  and  $f_1^{(b)}$ . (c) A Boolean network formed by selecting only  $f_1^{(a)}$  and  $f_2^{(b)}$ .

The chapter finishes with the computational results of improved methodology for modeling PBNs.

## 5.1 Problem formulation

A PBN is defined by the triplet  $(V, F, C)$ , where  $V = \{v_1, v_2, \dots, v_N\}$  is the set of Boolean variables. Each variable  $v_i$  is described by a set of Boolean functions,  $F_i = \{f_1^{(i)}, f_2^{(i)}, \dots, f_{|F_i|}^{(i)}\}$ . Each function  $f_j^{(i)}$  has a probability (or chance of selection) associated with it, which is given by the real number  $c_j^{(i)}$ . Using this terminology,  $F = \{F_1, F_2, \dots, F_N\}$  and  $C = \{C_1, C_2, \dots, C_N\}$ , where  $C_i = \{c_1^{(i)}, c_2^{(i)}, \dots, c_{|F_i|}^{(i)}\}$  such that  $\sum_{j=1}^{|F_i|} c_j^{(i)} = 1$ . A sample PBN is shown in Figure 5.1(a). In this figure, each variable can be described by two functions.

When a PBN  $(V, F, C)$  is used to represent a GRN,  $V = \{v_1, v_2, \dots, v_N\}$  correspond to the genes (or proteins) where  $N$  is the number of genes (or proteins) in the GRN. For each  $v_i$ , a corresponding set of Boolean functions  $F_i$  represent the Boolean relation between  $v_i$  and the genes that can have an influence on  $v_i$ . The probability values  $c_j^{(i)}$  represent the confidence on using the function  $f_j^{(i)}$  to explain the dynamics of  $v_i$ . PBN of Figure 5.1(a) correspond to a GRN of Figure 5.2.

A particular realisation of the PBN is given by the vector  $\mathbf{f} = \{f^{(1)}, f^{(2)}, \dots, f^{(N)}\}$  taking values in  $F_1 \times F_2 \times \dots \times F_N$ . By this definition, the number of possible realisations  $N_R$  of a PBN is given by Equation 5.1 :

$$N_R = \prod_{i=1}^N |F_i| \quad (5.1)$$

Each realisation of a PBN represents a Boolean network. All possible realisations of a PBN can be represented by using an  $N_R \times N$  matrix  $\mathbf{K}$  where every row  $i$  represent one of the possible network realisations of PBN given by the vector  $\mathbf{f}_i$ . Then the probability  $P_i$  that a network realisation  $i$  is selected





**Figure 5.2:** A Gene Regulatory Network corresponding to PBN in Figure 5.1(a).

$v_a v_b$	$f_1^{(a)}$	$f_2^{(a)}$	$f_1^{(b)}$	$f_2^{(b)}$
00	0	1	0	0
01	0	0	0	1
11	1	0	0	0
10	0	0	1	0

**Table 5.1:** Boolean functions corresponding to PBN in Figure 5.1(a).

is given by Equation 5.2, where  $c_{K_{ij}}^{(j)}$  defines the probability of choosing the function defined by  $k_{ij}$  for the gene  $v_j$ .

$$P_i = \prod_{j=1}^N c_{k_{ij}}^{(j)} \quad (5.2)$$

**Example 5.1.1** The PBN in Figure 5.1(a) is defined by the triplet  $(V, F, C)$  where  $V = \{a, b\}$ ,  $F = \{\{f_1^{(a)}, f_2^{(a)}\}, \{f_1^{(b)}, f_2^{(b)}\}\}$  and  $C = \{\{c_1^{(a)}, c_2^{(a)}\}, \{c_1^{(b)}, c_2^{(b)}\}\}$ . The Boolean functions in  $F$  are defined as in Table 5.1.

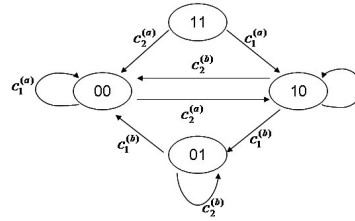
The given PBN can have four possible realisations ( $N_R = 2 \cdot 2$ ), given by the matrix  $\mathbf{K}$  in Equation 5.3, where the columns of  $\mathbf{K}$  correspond to the nodes  $a$  and  $b$  and rows correspond to different PBN realisations.

$$K = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{bmatrix} \quad (5.3)$$

The first row of matrix  $\mathbf{K}$  defines the PBN realization vector  $\mathbf{f}_1 = \{f_1^{(a)}, f_1^{(b)}\}$  (Figure 5.1(b)), the second row defines the realization vector  $\mathbf{f}_2 = \{f_1^{(a)}, f_2^{(b)}\}$  (Figure 5.1(c)) and so on. The probability of selecting the PBN realization  $\mathbf{f}_1$  is given by  $P_1 = c_1^{(a)} \cdot c_1^{(b)}$ . ■

Let us define a Boolean vector  $\mathbf{x}^t$  of size  $N$ , that represents the state of the network at time  $t$ . Gene  $v_i$  is **ON** or active at time  $t$  if  $x_i^t = 1$  and the gene  $v_i$  is **OFF** or inactive if  $x_i^t = 0$ . The probability of making a transition from a state  $\mathbf{x}^t$  to state  $\mathbf{x}^{t+1}$  can be computed by using Equation 5.4 [65].

$$P(\mathbf{x}^t, \mathbf{x}^{t+1}) = \sum_{i: f_{K_{i1}}^{(1)} = x_1^{t+1}, \dots, f_{K_{iN}}^{(N)} = x_N^{t+1}} P_i \quad (5.4)$$



**Figure 5.3:** State Transition Diagram corresponding to Figure 5.1(a). States 00,01 and 10 form an attractor. Labels on the edges represent the probability of transition

**Example 5.1.2** For the PBN in Example 5.1.1, the probability of transition from the state  $\{a = 0, b = 0\}$  to the state  $\{a = 1, b = 0\}$  is given by :

$$\begin{aligned} P(00, 10) &= P_3 + P_4 \\ &= c_2^{(a)} \cdot c_1^{(b)} + c_2^{(a)} \cdot c_2^{(b)} \end{aligned}$$

In the above equation, the probability of transition  $P(00, 10)$  is computed using Table 5.1, Equation 5.3 and Equation 5.4. The state transition diagram representing probability of transition among all possible  $2^N$  states of Figure 5.1(a) is as shown in Figure 5.3. The values on the edges in Figure 5.3 represent the probability of transition computed using Equation 5.4. ■

The state transition diagram of PBNs can be modeled as *Markov Chains* comprising of  $2^N$  states and the state transition matrix  $\mathbf{A}$  given by Equation 5.5 [65].

$$A(\mathbf{x}^t, \mathbf{x}^{t+1}) = P(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (5.5)$$

Using the transition matrix  $\mathbf{A}$ , the *Power method* can be used to compute the steady state probability distribution. In this method, given an initial probability distribution vector  $\mathbf{x}^{(0)}$ , Equation 5.6 is iterated until the condition in Equation 5.7 is satisfied for some tolerance  $\varepsilon$ .

$$\mathbf{x}^{(k)} = \mathbf{A}\mathbf{x}^{(k-1)} \quad (5.6)$$

$$\|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\|_\infty < \varepsilon \quad (5.7)$$

Once the matrix  $\mathbf{A}$  has already been constructed for a given PBN, Equations 5.6 and 5.7 have been observed to converge in a few iterations [145]. However, the above formalism is not fully suitable for large GRNs due to the following reasons :

- The computational complexity of Equation 5.4 is  $O(N_R)$ , where  $N_R$  is the number of possible PBN realisations. If each of the  $N$  genes in the network has two possible functions, then Equation 5.1 implies  $N_R = 2^N$ . This in turn implies that Equation 5.4 can not be used for networks that have a very large number of genes ( i.e.  $N$ ) even if the number of alternate functions for each gene is as small as 2.

- Construction of the state transition matrix, as defined by [65], has the computational complexity of  $O(N_R \cdot 2^N \cdot 2^N)$ . Under synchronous transition assumption the matrix  $\mathbf{A}$  may have at most  $N_R \cdot 2^N$  non-zero entries [67]. Based on this observation, an improved algorithm was proposed in [145] that computes only those transitions that have non-zero probabilities and hence reduces the complexity of the construction of the matrix  $\mathbf{A}$  to  $O(N_R \cdot 2^N)$ . Even with this reduced complexity, the factor  $N_R$  can be very large for even simple networks as we described in the previous point. Even if  $N_R$  is small, an explicit construction of the transition matrix still has an exponential complexity.
- Computing a steady state distribution using Equations 5.6 and 5.7 depends upon the starting distribution  $\mathbf{x}^{(0)}$ . In the case of multiple attractors, different steady state probability distributions may be found depending upon the chosen  $\mathbf{x}^{(0)}$ . This is an issue because it is not possible to make a claim a priori on the number of attractors for a given network using this method. Further, since it is more interesting to study gene regulatory networks with multiple steady states (or cell states), the current methodology prevents the user from exploiting the full functionality of PBNs.

In the next section, we show how these issues are addressed by using the implicit representation and traversal techniques based on ROBDDs and ADDs.

## 5.2 Modified formulation

First we introduce a modified form of Equation 5.4 to reduce the complexity of computing the probability of transition  $P_i$  from  $O(N_R)$  to  $O(\tilde{N}_R)$ , where  $\tilde{N}_R$  is given by Equation 5.8.

$$\tilde{N}_R = \sum_{i=1}^N |F_i| \quad (5.8)$$

In Equation 5.8,  $\tilde{N}_R$  can also be seen as the number of Boolean functions in a given PBN. The modified formula for computing the probability of transition  $P(\mathbf{x}^t, \mathbf{x}^{t+1})$  is then given by Equations 5.9 and 5.10:

$$P_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \sum_{j: f_j^{(i)} = x_i^{t+1}} c_j^{(i)} \quad (5.9)$$

$$P(\mathbf{x}^t, \mathbf{x}^{t+1}) = \prod_{i=1}^N P_i(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (5.10)$$

In Equation 5.9,  $0 \leq P_i(x^t, x^{t+1}) \leq 1$  and it is equal to the sum of the probabilities associated with all those functions  $f_j^{(i)}$  for which the expression of

gene  $v_i$  at time  $t + 1$  matches the function evaluation (i.e.  $f_j^{(i)} = x_i^{t+1}$ ). The probability of transition of the state of the gene regulatory network from  $\mathbf{x}^t$  to  $\mathbf{x}^{t+1}$  is given by the product of the transition probabilities of all the genes and is defined by Equation 5.10. The computational complexity of Equation 5.10 is  $O(\tilde{N}_R)$ . To get an idea of the improvement in complexity with this modified equation, if each gene can have  $k$  functions then  $\tilde{N}_R = k \cdot n$ , whereas  $N = k^n$  in Equation 5.1. Mathematically, Equations 5.4 and 5.10 are equivalent.

However, using this modified equation may not help in reducing the complexity of constructing the transition matrix  $\mathbf{A}$  as the number of non-zero elements in the matrix are  $O(N_R \cdot 2^N)$  and using this modified equation for each non-zero entry would only increase the computational complexity to  $O(\tilde{N}_R \cdot N_R \cdot 2^N)$ . If the size of the transition matrix  $\mathbf{A}$  could be reduced so that the number of nonzero entries are significantly smaller than  $O(N_R \cdot 2^N)$ , then using this modified equation could be helpful. This is the central idea behind the ROBDD based technique proposed in this chapter. We compute the steady states (or attractors) using an implicit representation based on ROBDDs and compute the transition matrix (implicitly using ADDs) only for the states restricted to the attractors. This way the size of the matrix  $\mathbf{A}$  is proportional to the size of the attractor. Although it is difficult to put a bound on the number of states in an attractor, it has been observed that it is often a small number [145]. Dividing the problem of computing the steady state distribution into two parts ensures that in the event of a situation in which the size of an attractor becomes very large to be represented as ADDs, our algorithm is at least able to detect the states in the attractor. However, we may not be able to compute the probability distribution of these states in that situation.

## 5.2.1 Implicit formulation

We start by giving a modified formulation of the transition functions  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  and  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  given in Chapter 3. With the modified transition functions, algorithms proposed in Chapter 3 can be used for computing the attractors of the PBN formulation of GRNs.

Given a PBN, a gene expression  $x_i$  can have only two states ‘0’ and ‘1’. Since there are multiple functions  $f_j^{(i)}$  acting on the gene  $x_i$  at the same time, there is a possibility that some of the functions set the expression of the gene  $x_i$  to ‘0’ and some others may set the expression to ‘1’. The transition function  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  describing the relationship between the expression of a gene  $x_i$  at time  $t + 1$  and the expression of all the other genes in the GRN at time  $t$  is given in Equation 5.11.

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \underbrace{\{x_i^{t+1} \Leftrightarrow \bigvee_{j=1}^{|F_i|} f_j^{(i)}\}}_I \vee \underbrace{\{x_i^{t+1} \Leftrightarrow \bigwedge_{j=1}^{|F_i|} f_j^{(i)}\}}_{II} \quad (5.11)$$

Part (I) of Equation 5.11 represents the situation when one of the input func-

tions  $f_j^{(i)}$  can set the expression of gene  $v_i$  (given by  $x_i^{t+1}$ ) to 1. Part (II) of Equation 5.11 represents the case when one of the input functions  $f_j^{(i)}$  can set  $x_i^{t+1}$  to 0.

If all the genes in the network are assumed to make a synchronous transition then the transition function representing the state of the network between consecutive time steps can be given by Equation 5.12 :

$$T(\mathbf{x}^t, \mathbf{x}^{t+1}) = \bigwedge_{i=1}^N T_i(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (5.12)$$

Equations 5.11 and 5.12 are the counterpart of Equations 3.3 and 3.4 respectively, proposed for synchronous Boolean modeling in Chapter 3. With the modified transition functions, Algorithm 2 proposed in Chapter 3 can be used for computing all the attractors. Now for every attractor, we can construct the matrix  $A(\mathbf{x}^t, \mathbf{x}^{t+1})$  restricted to the states in that attractor and compute the steady state probability distribution for those states.

Equation 5.9 can also be computed symbolically using Equations 5.13 and 5.14.

$$q_j^{(i)} = (x_i^{t+1} \Leftrightarrow f_j^{(i)}) \cdot c_j^{(i)} \quad (5.13)$$

$$\tilde{P}_i(\mathbf{x}^t, \mathbf{x}^{t+1}) = \bigvee_{j=1}^{|F_i|} q_j^{(i)} \quad (5.14)$$

The probability of transition of the state of the network is then given by Equation 5.15.

$$\tilde{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \bigwedge_{i=1}^N \tilde{P}_i(\mathbf{x}^t, \mathbf{x}^{t+1}) \quad (5.15)$$

Equations 5.14 and 5.15 can be computed implicitly using the algorithms given in the next section.

## 5.3 PBN Functionalities

PBNs can be used for performing various interesting computational analysis. For instance, computation of steady states, computation of the probability of a path from one state of the network to another state and identification of genes in a GRN that play a most important (or least important) role in the dynamics of the PBN. These computational tasks have been addressed in the literature [65, 67, 145]. However, existing tools use an explicit representation and modeling of PBNs, restricting their application to networks with no more than 10 genes. In this section, we will give the algorithms to perform above functionalities in an efficient manner on the implicit formalism of PBNs introduced in the previous section.

---

**Algorithm 6:** Algorithm for computing the stationary distribution of states in an attractor.

---

```

1 comp_stationary_distribution( $T, \tilde{P}, \epsilon$ )
2 begin
3    $SS = \text{all\_attractors}(T)$ 
4   for  $i = 1$  to  $|SS|$  do
5      $S^{src} = SS_i$ 
6      $S^{dest} = SS_i(x^t \leftarrow x^{t+1})$ 
7      $\hat{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \tilde{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) \wedge S^{src} \wedge S^{dest}$ 
8      $q^{(0)} = \text{init\_vector}(S^{src}, S^{src})$ 
9      $k = 0$ 
10    while true do
11       $q^{(k+1)} = \backslash_{x_t}^+ [\hat{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) \cdot q^{(k)}]$ 
12      if  $|\backslash_{x_t}^{Max} q^{(k+1)} - \backslash_{x_t}^{Max} q^{(k)}| \leq \epsilon$  then
13         $p^{SS_i} = q^{(k+1)}$ 
14        break;
15       $k = k + 1$ 
16    return  $p^{SS}$ 
17 end
```

---

### 5.3.1 Algorithm for computing steady state distribution

Algorithm 6 formally describes the procedure for computing the stationary distribution of the attractors in a PBN. Algorithm 6 takes as input the transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  given in Equation 5.12, the transition probability function  $\tilde{P}(\mathbf{x}^t, \mathbf{x}^{t+1})$  given in Equation 5.15 and the convergence factor  $\epsilon$ . In Line 3 of Algorithm 6, we compute all the attractors that exist in the state space using the function `all_attractors()` that was defined in Algorithm 2 of Chapter 3. Then for every attractor identified in Line 3, the state space of probability transition function is restricted to the states in that attractor (Line 7). The power method (i.e. Equations 5.6-5.7) for computing the stationary distribution is executed in Lines 8-15. In Line 8, an initial distribution is computed using the function `init_vector( $\tilde{S}, S$ )`, which selects a random state from the set of states  $\tilde{S}$  and assigns it a probability 1. All the remaining states in the set  $S$  are assigned the probability 0. Then starting from this initial distribution, the probability distribution for the next iteration of the power method is computed in Line 11. The operation  $\backslash_{x_t}^+$  in Line 11 stands for arithmetic existential quantification (defined in Chapter 2 in more detail). Then we check for convergence of distribution in Line 12 (implements Equation 5.7) and the iterations are terminated if the distribution has converged (Lines 13-14).

**Algorithm 7:** Computing Forward and Backward reachable sets

---

```

1 forward_set( $S_0, T, S^{target}$ )
2 /* backward_set( $S^0, T, S^{target}$ ) */
3 begin
4    $RS^{(0)} \leftarrow \emptyset, FS^{(0)} \leftarrow \{S^0\}$ 
5    $k \leftarrow 0$ 
6   while  $FS^{(k)} \neq \emptyset$  do
7      $FS^{(k+1)} = I_f(FS^{(k)})(x^{t+1} \leftarrow x^t) \wedge \overline{RS^{(k)}}$ 
8     /*  $FS^{(k+1)} = I_b(FS^{(k)})(x^t \leftarrow x^{t+1}) \wedge \overline{RS^{(k)}}$  */
9      $RS^{(k+1)} = RS^{(k)} \vee FS^{(k+1)}$ 
10    if  $RS^{(k+1)} \wedge S^{target} \neq \emptyset$  then
11      return ( $RS^{(k+1)}$ )
12     $k \leftarrow k + 1$ 
13  /* Target is not reachable from source. Return Empty Set */
14  return  $\emptyset$ 
15 end

```

---

**5.3.2 Probability of a path**

Given a path  $p(i \rightsquigarrow^1 j) = (i, s_1, s_2, \dots, s_n, j)$  from state  $i$  to state  $j$ , the probability of this path,  $P(i \rightsquigarrow^1 j)$  is given by Equation 5.16 :

$$P(i \rightsquigarrow^1 j) = P(i, s_1) \cdot P(s_1, s_2) \cdot \dots \cdot P(s_n, j) \quad (5.16)$$

There can be multiple paths between any two states and the probability  $P(i \rightsquigarrow j)$  to go from a state  $i$  to state  $j$  is given by the sum of the probability of all these paths. Summation of the probability of all the paths  $l$  maintains the probability constraint, i.e.  $0 \leq \sum_l P(i \rightsquigarrow^l j) \leq 1$ . To avoid making too many  $P(i \rightsquigarrow^l j)$  computations, we apply a constraint that only the shortest paths (or paths with minimum number of steps) are enumerated between any two states. This gives a lower bound on the probability  $P(i \rightsquigarrow j)$ .

Given the state transition matrix  $\mathbf{A}$ , a row vector  $\mathbf{y}^{(0)}$  of size equal to the number of states in the state transition diagram is constructed. In this vector  $\mathbf{y}^{(0)}$  all the entries are 0, except the entry corresponding to the source state which is 1. Then Equation 5.17 is iterated until the condition in Equation 5.18 is satisfied.

$$\mathbf{y}^{(k)} = \mathbf{A}\mathbf{y}^{(k-1)} \quad (5.17)$$

$$y^{(k)}[dest] > 0 \quad (5.18)$$

Intuitively, Equation 5.17 computes the probability of reaching a state  $i$  in  $k$  iterations. The probability values are stored in the vector  $\mathbf{y}^{(k)}$ . Equation 5.18 ensures that we terminate the iterations the first time we reach the destination state.

Equations 5.17 and 5.18 have similar computational issues as the probability distribution computation using Equations 5.6 and 5.7 in Section 5.3.1. But

---

**Algorithm 8:** Computing set of states that lie on all shortest paths between a source and destination

---

```

1 compute_states_path( $T, S^{src}, S^{dest}$ )
2 begin
3    $FR = forward\_set(S^{src}, T, S^{target})$ 
4    $S^{src\_tmp} = S^{src}(x^t \leftarrow x^{t+1})$ 
5    $S^{dest\_tmp} = S^{dest}(x^t \leftarrow x^{t+1})$ 
6    $BR = backward\_set(S^{dest\_tmp}, T, S^{src\_tmp})$ 
7    $S^{Reduced} = FR \wedge BR(x^t \leftarrow x^{t+1})$ 
8   return  $S^{Reduced}$ 
9 end
```

---



---

**Algorithm 9:** Algorithm for computing probability of transition from a source state to a destination state in one or more transitions.

---

```

1 tran_matrix_path( $T, \tilde{P}, s^{src}, s^{dest}$ )
2 begin
3    $S = compute\_states\_path(T, s^{src}, s^{dest})$ 
4   if  $S = \emptyset$  then
5     /* Target is not reachable from source. Return NULL */
6     return  $NULL$ 
7    $S^{src} = S$ 
8    $S^{dest} = S(x^t \leftarrow x^{t+1})$ 
9    $\hat{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) = \tilde{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) \wedge S^{src} \wedge S^{dest}$ 
10   $q^{(0)} = init\_vector(s^{src}, S)$ 
11   $k = 0$ 
12  while true do
13     $q^{(k+1)} = \setminus_{x_t}^+ [\hat{P}(\mathbf{x}^t, \mathbf{x}^{t+1}) \cdot q^{(k)}]$ 
14     $P(s^{src} \rightsquigarrow s^{dest}) = \setminus_{x_t}^{Max} [q^{(k+1)} \wedge s^{dest}]$ 
15    if  $P(s^{src} \rightsquigarrow s^{dest}) > 0$  then
16      return  $P(s^{src} \rightsquigarrow s^{dest})$ 
17     $k = k + 1$ 
18 end
```

---

again, if a smaller transition matrix  $\mathbf{A}$  can be used then  $P(src \rightsquigarrow dest)$  can be computed very efficiently. Using the implicit representation we described earlier, here we propose algorithms that first compute the states that lie on all the shortest paths between the source and the destination and then compute  $\tilde{\mathbf{A}}$  restricted to these states.

Algorithms 7, 8 and 9 can be used to compute  $P(i \rightsquigarrow j)$ . Algorithms 7 and 8 compute the states that lie on the shortest paths between the source and the destination states. In Algorithm 8, first the forward reachable states from the source state are computed in Line 3. Then the backward reachable states from the destination state are computed in line 6. The intersection of the forward and the backward reachable states gives all the states that may lie on all the paths between the source and the destination. To ensure that we include only



the shortest paths while computing the forward and backward reachable states in Algorithm 7, we compute reachable states only up to the point when the target state is first seen (in line 10). Then Algorithm 9 restricts the state space of the probability transition matrix to the states that lie on the shortest paths in Line 9. The function `init_vector()` assigns the probability 1.0 to the state  $s^{src}$  and the probability 0 to all the other states that lie on the shortest paths between the source state and the destination state. Then Equations 5.17 and 5.18 are implemented as in Lines 10-17. In every iteration of the *While loop* of Lines 12-17, the probability distribution in next step is computed in Line 13. Then in Line 14, we compute the probability of reaching the destination state. If this probability is greater than zero then the iterations are terminated and the resulting probability from the source state to the destination state is returned.

### 5.3.3 Algorithm for sensitivity analysis

Another interesting functionality of PBNs is their ability to represent the influence of a gene on other genes in the network. This functionality can be very useful in deciding the genes that should be perturbed to have the maximum impact on the state space of the network. This impact could be in terms of the size of the attractor, number of attractors or the probability of the reachability from one state to another. The influence of a variable  $x_i$  on a function  $f(x_1, x_2, \dots, x_n)$  is given by Equations 5.19-5.21 [65], where  $\frac{\partial f(\mathbf{x})}{\partial x_i}$  is the Boolean difference of Function  $f(\mathbf{x})$  with respect to the variable  $x_i$ .

$$I_i(f) = Pr \left\{ \frac{\partial f(\mathbf{x})}{\partial x_i} = 1 \right\} \quad (5.19)$$

$$= Pr \{ f(x_i = 0) \neq f(x_i = 1) \} \quad (5.20)$$

$$= \frac{\text{no. of } \mathbf{x} \text{ such that } \frac{\partial f(\mathbf{x})}{\partial x_i} = 1}{2^{\text{size of vector } \mathbf{x}}} \quad (5.21)$$

Equation 5.22 can then be used to compute the influence of a gene  $v_i$  on the gene  $v_j$ .

$$I_i(v_j) = \sum_{k=1}^{|F_i|} I_i(f_k^{(j)}) \cdot c_k^{(j)} \quad (5.22)$$

This way an  $N \times N$  *influence matrix*  $\Gamma$  can be constructed to represent the influence of every gene on all the other genes.

In our implicit representation of PBNs, Equation 5.21 can be trivially computed by enumerating cubes of the Boolean function  $\frac{\partial f(\mathbf{x})}{\partial x_i}$ . This can be better demonstrated with Example 5.3.1.

**Algorithm 10:** Algorithm to compute Influence Matrix

---

```

1 compute_influence_matrix( $F, C$ )
2 begin
3   for  $i = 1$  to  $N$  do
4     for  $j = 1$  to  $N$  do
5       for  $k = 1$  to  $|F_i|$  do
6          $\frac{\partial f}{\partial x} = \exists_{x_i}(f_k^{(j)} \wedge \neg x_i) \oplus \exists_{x_i}(f_k^{(j)} \wedge x_i)$ 
7         cubes_drv = compute_Cubes( $\frac{\partial f}{\partial x}$ )
8         cnt = 0
9         for  $p = 1$  to cubes_drv.size() do
10          cnt +=  $2^{n - \text{num\_var\_support}(\text{cubes\_drv}[p])}$ 
11           $\Gamma_{ij} += \frac{\text{cnt}}{2^n} \times c_k^{(j)}$ 
12 end

```

---

**Example 5.3.1** Given a Boolean function  $f(a, b, c)$  in Equation 5.23, it can be rewritten in a *sum of product* SOP form as in Equation 5.24.

$$f = (a \oplus b)c + bc \quad (5.23)$$

$$f = \bar{a}\bar{b}c + bc \quad (5.24)$$

In Equation 5.24, a three variable function  $f$  consists of two cubes  $\bar{a}\bar{b}c$  and  $bc$ . Cube  $\bar{a}\bar{b}c$  has all the three variables in support and evaluates to true for only one ( $= 2^{3-3}$ ) Boolean vector (i.e.  $a = 0, b = 0, c = 1$ ). Cube  $bc$  has two variables in support and evaluates to true for two ( $= 2^{3-2}$ ) Boolean vectors  $\{111, 110\}$ . These two cubes together give three Boolean vectors  $\{001, 111, 110\}$  for which function  $f$  can be true. ■

In a ROBDD representation of a Boolean function, any path from root node to **1**-leaf node represents a cube. So computing cubes is very efficient in BDDs and the number of cubes is always less than the number of possible Boolean vectors. Based on cube enumeration, the method for computing the influence matrix is formally described in Algorithm 10. In this algorithm, an  $N \times N$  influence matrix  $\Gamma$  is constructed in lines 3-11. The influence of gene  $v_i$  on  $v_j$  (i.e.  $\Gamma_{ij}$ ) is computed in lines 5-11. A *For loop* is iterated (in Lines 5-11) over all the functions that may influence the gene  $v_j$ . For each function, the Boolean difference is computed in line 6. Then the cubes of the Boolean difference are stored in an array in line 7. In lines 9-10, the number of Boolean vectors for all the cubes are computed and the corresponding influence for this function is finally added to  $\Gamma_{ij}$  in line 11.

## 5.4 Results

In this section, we present the results obtained from the simulations on the glioma network [49, 67]. Based on the glioma gene expression data set of

**Table 5.2:** Boolean functions in glioma network.

Gene Id	Gene Name	Input Functions		
1	Tie-2	$f_1^{(1)}(2, 13)$	$f_2^{(1)}(7, 14)$	
2	TGF-beta3	$f_1^{(2)}(1, 11, 12)$	$f_2^{(2)}(10, 13)$	
3	ERCC1	$f_1^{(3)}(2, 6)$	$f_2^{(3)}(4, 10, 11)$	
4	HSP40	$f_1^{(4)}(1, 3)$		
5	TDPX2	$f_1^{(5)}(1, 3)$		
6	GSTP1	$f_1^{(6)}(1, 2)$	$f_2^{(6)}(3)$	$f_3^{(6)}(9, 10, 8)$
7	GNB1	$f_1^{(7)}(6)$		
8	NDPKB	$f_1^{(8)}(6, 7)$	$f_2^{(8)}(9, 10, 11)$	
9	SCYB10	$f_1^{(9)}(6, 10, 8)$		
10	PDGFA	$f_1^{(10)}(9, 3, 11)$		
11	NKEFB	$f_1^{(11)}(8, 10)$	$f_2^{(11)}(12)$	
12	Beta Actin	$f_1^{(12)}(11)$		
13	NFKB1	$f_1^{(13)}(1, 2, 11)$	$f_2^{(13)}(12, 14)$	
14	BCL2A1	$f_1^{(14)}(1, 4)$	$f_2^{(14)}(12, 8, 13)$	

[49], a small PBN of 14 genes was proposed in [67]. The 14 genes selected in the Glioma Network (Table 5.2) play a very important role in the formation of blood vessels. Presence or absence of some of them can be used to differentiate between a healthy cell and a tumor cell. For example, Tyrosine Kinase receptors (Tie-2) along with angiopoietins plays an important role in vasculogenesis [127] (formation of blood vessels), the *excision repair cross-complementing* (ERCC1) gene helps in DNA damage repair, while mutations in *Nuclear Factor-Kappa B* (NFkB) has been linked to the formation of tumors. On modeling this network, we find a single attractor consisting of 4450 states. Table 5.4 gives the number and size of attractors when different genes are perturbed in the glioma network. A gene is knocked out when it is constantly inactive (i.e. level 0) and it is over-expressed when it is constantly active (i.e. level 1). Table 5.4 represents the situation when some of the functions are inactive in the glioma network. Function  $f_j^{(i)}$  in Table 5.4 represents the  $j^{th}$  function of gene  $v_i$ . Further details of Functions  $f_j^{(i)}$  are given in Tables 5.2 and 5.3.

All the results obtained from simulations on the glioma network match the results shown in [145]. The computation time of our algorithm was, at maximum, 65 seconds on a 1.8 GHz Dual Core Pentium machine with 1GB of RAM running on Linux Fedora Core 5. The algorithms are implemented in C++ using the CUDD package for BDDs and ADDs. It is difficult to compare the run time of our algorithms with the one achieved by the authors in [145] since their program is not available in the PBN toolbox [66]. However, it has been mentioned in their paper that it takes them 20 minutes to compute the steady state distribution on a CPU Pentium 4 machine with 1GB RAM.

**Table 5.3:** Truth table for input functions.

Input Vector	$f_1^{(1)}$	$f_2^{(1)}$	$f_1^{(2)}$	$f_2^{(2)}$	$f_1^{(3)}$	$f_2^{(3)}$	$f_1^{(4)}$	$f_1^{(5)}$	$f_1^{(6)}$	$f_2^{(6)}$	$f_3^{(6)}$	$f_1^{(7)}$
000	0	1	1	0	0	0	0	1	1	0	0	1
001	1	0	1	1	1	1	1	0	0	1	0	-
010	1	1	0	1	1	0	-	-	1	-	1	-
011	0	0	0	1	1	0	-	-	1	-	1	-
100	-	-	1	-	-	0	-	-	-	-	0	-
101	-	-	0	-	-	1	-	-	-	-	1	-
110	-	-	1	-	-	1	-	-	-	-	0	-
111	-	-	0	-	-	0	-	-	-	-	1	-

Input Vector	$f_1^{(8)}$	$f_2^{(8)}$	$f_1^{(9)}$	$f_1^{(10)}$	$f_1^{(11)}$	$f_2^{(11)}$	$f_1^{(12)}$	$f_1^{(13)}$	$f_2^{(13)}$	$f_1^{(14)}$	$f_2^{(14)}$
000	1	1	0	1	1	0	1	0	1	1	0
001	0	1	0	1	0	1	0	1	0	1	0
010	0	0	1	0	0	-	-	1	0	1	1
011	0	0	1	0	1	-	-	0	1	0	0
100	-	0	1	0	-	-	-	1	-	-	1
101	-	0	0	1	-	-	-	1	-	-	1
110	-	1	1	0	-	-	-	1	-	-	0
111	-	0	1	0	-	-	-	0	-	-	0

**Table 5.4:** Gene perturbation simulation

Gene Perturbed	Number Of		Time (sec)
	attractors	states	
Un-perturbed	1	4450	50
ERCC1 = 1	1	768	6
SCYB10 = 0	1	1920	33
NFkB1 = 0	1	2145	38

**Table 5.5:** Influence of Functions on Attractors.

Inactive Function	Number Of		Time (sec)
	attractors	states	
$f_2^{(1)}$	1	4424	38
$f_1^{(3)}$	1	1633	27
$f_2^{(14)}$	1	3815	36

**Table 5.6:** Computational Results on Synthetic Data.

Genes	Number of			Attractors	Maximum Attractor size	Time (sec)	
	Alt. Functions					Steady States	Probability Distribution
	1	2	3				
20	10	10	0	2	844	0.6	8
20	9	8	3	3	288	0.8	3
40	30	10	0	8	1536	2	7
40	23	15	2	3	2684	3	30
60	41	19	0	12	4096	39	40
60	46	10	4	18	1536	120	20

Moreover, their software is written in MATLAB. The PBN toolbox in [66] can not run networks that have more than 10 nodes as it quickly runs out of memory. Even with a network of 10 nodes and two functions per gene, it takes more than 1 hour to compute the state transition matrix.

Some results to benchmark our algorithm are given in Table 5.6. Columns 2, 3 and 4 represent the number of genes in the network with 1, 2 and 3 alternate functions per gene respectively. For most of the synthetic PBNs in Table 5.6, more than one attractor can exist. Using the existing PBN toolbox [66] it is not possible to simulate such networks as there will be no unique steady state distribution. However, using algorithms proposed in this chapter,

one can easily identify different attractors in the state space and compute the stationary distribution of each attractor individually.

## 5.5 Summary

In this chapter, we proposed an extension of Boolean modeling formalism proposed in Chapter 3 to PBNs. In practice, PBNs can be very helpful in analysing the experimental data. Since the data coming from experimental tools is always noisy, it is often possible to have more than one explanation for the same observation. If the inference of underlying gene regulatory network is the final aim of data analysis task, then multiple inferred functions can be directly translated into PBNs and the dynamic analysis of resulting networks can be performed. In the literature, the potential applications of PBN have been restricted by the unavailability of efficient computational tools to model large GRNs. PBN toolbox proposed in the literature use explicit methods for constructing the state transition probability matrix and model it using computational methods developed for Markov chains. However, the resulting Markov chain is not guaranteed to have unique stationary distribution due to the presence of multiple attractors. Algorithms proposed in this chapter can address this issue by dividing the problem of computing stationary distribution of attractors into: attractor computation and then computing the stationary distribution of the states in all the attractors individually. Additionally, implicit representation using ROBDDs and ADDs ensure that even large GRNs can be modeled in a realistic run time.

PBNs model the stochasticity arising due to a choice between alternate biological functions for activating a given gene (or protein) in a GRN. However, stochasticity in a GRN can also be induced due to inherent noise in the biological function underlying the regulation of a gene (or protein). To address the stochasticity arising from temporary function failure, we introduce in the next chapter an extension of Boolean formalism which can also be generalised to PBN.



---

# Stochasticity in Gene Regulatory Networks

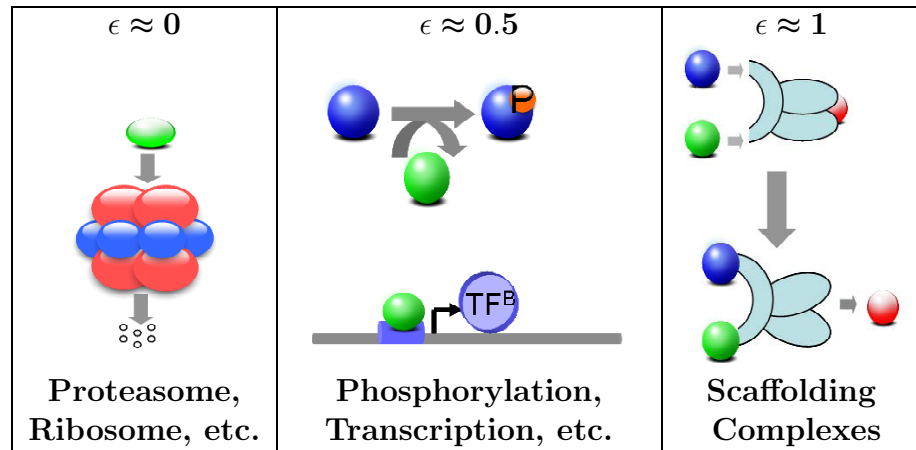
---

# 6

Biological functions can have varying levels of complexity and hence, show varying levels of stochasticity in their behaviour. Although it is experimentally difficult to quantify the measure of stochasticity involved in different biological functions, it is a well known fact that some functions, such as proteasome degradation, are least prone to stochasticity while functions, such as scaffolding complexes that integrate signals arising from different pathways, are likely to behave most stochastically. In practice, most biological functions behave somewhere between the above two extremes. Keeping this observation in mind, the probability of stochasticity can be broadly classified into three different classes, namely: low probability of error ( $\epsilon \approx 0$ ), medium probability of error ( $\epsilon \approx 0.5$ ) and high probability of error ( $\epsilon \approx 1$ ). Figure 6.1 gives an example of a few biological functions divided into these different classes of stochasticity.

Traditionally, the stochasticity in Boolean models of GRNs is modeled by flipping the expression of nodes in a GRN from 0 to 1 or vice versa with some predefined flip probability [14, 90, 39, 114]. This model of stochasticity is referred to as *stochasticity in node* (SIN) in this chapter. However, the SIN model of stochasticity does not take into account the complexity of underlying biological function while flipping the expression of a node.

This chapter introduces a new stochasticity modeling approach called the *stochasticity in functions* (SIF) for Boolean models of GRNs. In the SIF model, stochasticity is induced at the level of biological functions rather than at the level of expression of a protein/gene. SIF associates a probability of failure with different biological functions and models stochasticity in these functions depending upon the expression of the input nodes. With the above two constraints in the SIF model, the probability of a node in the GRN flipping its expression from 0-to-1 or vice-versa at a given time instant depends upon the probability of function failure and the activity of other nodes in the network at

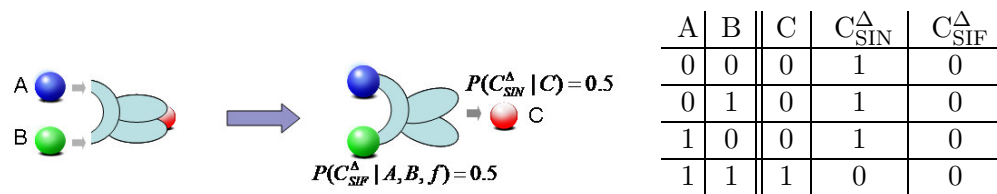


**Figure 6.1:** Biological functions categorized into three different classes of stochasticity and error probability. From left to right, biological processes are classified from very stable structures to highly stochastic systems involving scaffold proteins.

that instant in time, thereby making it possible to integrate the stochasticity due to complexity of a biological function with the dynamics of the GRN.

**Example 6.0.1** A difference between the SIN and SIF models of stochasticity can be better understood from Figure 6.2. In Figure 6.2, we model a biological function that requires simultaneous activity of inputs  $A$  and  $B$  to activate the protein  $C$ . Under the SIN model, probability of flipping the expression of node  $C$  is given by  $P(C_{SIN}^\Delta | C) = 0.5$ . While under the SIF model, probability that node  $C$  shows a noisy behaviour is given by  $P(C_{SIF}^\Delta | A, B, f)$ , which is a function of the biological phenomena  $f$  and its inputs  $\{A, B\}$ . The table next to the figure represents the stochastic expression of  $C$  under the SIN and the SIF model. ■

The chapter starts with an introduction to impact of stochasticity on the dynamics of a GRN in Section 6.1. Then we formally introduce the SIN and the SIF models of stochasticity in Section 6.2. Finally, the results on applying the two models of stochasticity on T-helper and T-cell receptor networks are compared and contrasted in Section 6.3.



**Figure 6.2:** A small example demonstrating the difference between the SIN and SIF in terms of the activity of the output gene (or protein).



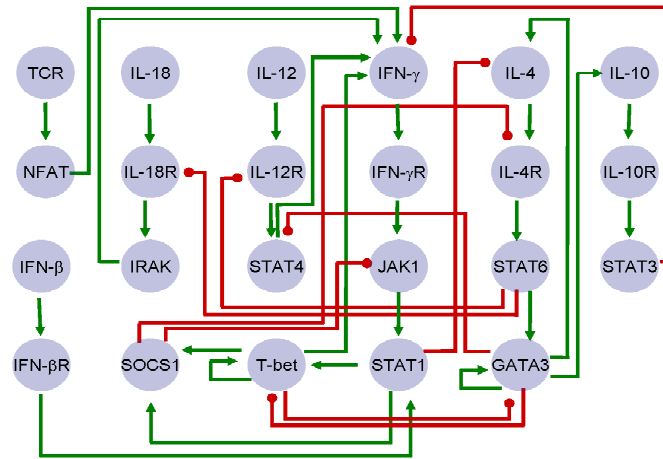
## 6.1 Impact of Stochasticity

In modeling GRNs, one is often more interested in knowing the steady state behavior of the network as compared to that of the transient states. This is because in biological experiments, measurement and comparison of the transient states across multiple experiments is difficult as the dynamics may vary in each experiment. Steady state behavior, which corresponds to the end point of an experiment when all cells stabilise, is easier to measure and compare with similar experiments. It is also experimentally easier to understand the impact of stochasticity on the steady state behavior (by measuring the fraction of different cell phenotypes in an experiment). In the dynamic simulation of GRNs, a state of the network evolves over time and stabilises in an *attractor* (or the steady state). Hence, an attractor represents the long term behaviour of the genes/proteins in the regulatory networks. Attractors (or the steady states) of Boolean networks are hypothesized to correspond to the cellular steady states (or phenotypes) [143, 139, 56]. In this chapter, we compare the results obtained using SIN and SIF stochasticity models with respect to two properties of steady states: (a) Cellular Differentiation in response to an external stimulus and (b) Robustness of attractors.

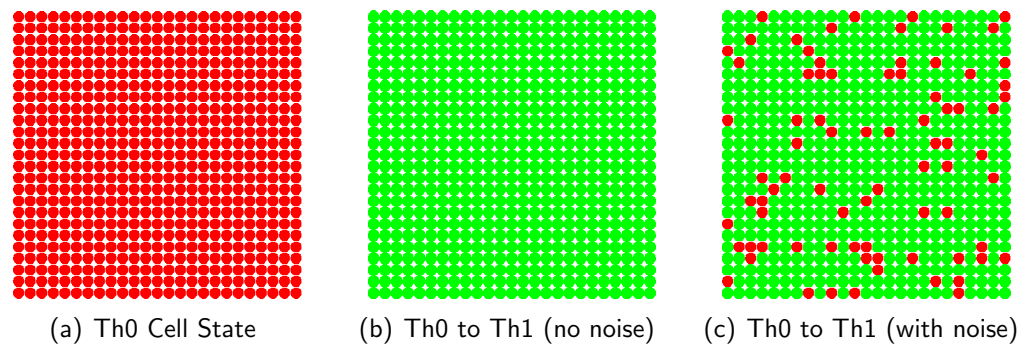
### 6.1.1 Cellular differentiation

In the absence of stochasticity, all biological functions behave as per their description and an initial state of the network differentiates into a specific steady state. However, in the presence of stochasticity in these functions, a network simulation starting from the same initial state may stabilise into different steady states. The probability of differentiating into one steady state can be different from the probability of differentiating into another steady state. This simulation behavior can be used to explain the well known biological observation of emergence of phenotypically distinct subgroups within an isogenic [111] cell population in response to an input stimuli (such as on exposure to external ligands). A sample simulation experiment on a T-Helper Differentiation network [102], as in Figure 6.3, can be effectively used to describe the stochastic differentiation of naïve T-Helper cells (i.e., Th0) in response to a pulse of  $\text{IFN}\gamma$ , a key cytokine known to play an important role in Th0 to Th1 differentiation.

In Figure 6.4(a), cells are initially in a naïve undifferentiated cell state (i.e., Th0). On receiving an input stimulus on  $\text{IFN}\gamma$ , cells must differentiate into Th1 cell state in the absence of any stochasticity. This is shown in Figure 6.4(b). Biologically it is known that, while most of the cells should differentiate into Th1 state in response to an  $\text{IFN}\gamma$  dosage, a few cells can revert to the Th0 state [19, 94]. This difference in response across the cell population is often attributed to inherent stochasticity in biological functions (Figure 6.4(c)).



**Figure 6.3:** T-Helper Gene Regulatory Network [102].



**Figure 6.4:** Simulation results showing the effect of noise on T-helper cell differentiation process with an external stimulus of IFN $\gamma$ . Each small circle is representative of a T-Helper cell and each cell is modeled to behave independent of the neighbouring cells. Red cells represent the naïve undifferentiated Th0 cells, green cells represent Th1 cell state. Ratio of number of red (or green) cells to total number of cells in a panel is representative of the probability of differentiating into Th0 (Th1 or Th2) cell state. (a) Cell Culture maintained in Th0 state. (b) In absence of any stochasticity all Th0 cells differentiate to Th1 cell state on receiving IFN $\gamma$ . (c) In the presence of stochasticity, Th0 cells differentiate into Th1 cells while some cells cannot differentiate on receiving IFN $\gamma$  and revert to Th0 cell state.

### 6.1.2 Robustness of attractors

Robustness of attractors of a GRN can be defined as the probability of an attractor reverting back to itself when the expression of one or more nodes is perturbed from its original expression value. In the absence of any stochasticity in the biological functions, there should be no transition among two different attractors. If a perturbation changes the state of an attractor, it is possible that the new perturbed state may transition into a different attractor. The perturbed state may be generated in response to external stimuli

A	C	$C^\Delta$
0	0	1
1	1	0

A	C	$C^\Delta$
0	1	0
1	0	1

A	B	C	$C^\Delta$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	C	$C^\Delta$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

A	B	C	$C^\Delta$
0	0	0	1
0	1	0	1
1	0	1	0
1	1	0	1

(a) BUFF      (b) NOT      (c) AND      (d) OR      (e) IAND

**Table 6.1:** Truth tables representing the transfer function of different Boolean logic gates.  $A$  and  $B$  are the input genes,  $C$  represents the output gene expression in the absence of stochasticity and  $C^\Delta$  represents the output gene expression in the presence of stochasticity.

such as ligands, inhibitors or due to internal stochasticity of the cell. Biologically, cellular steady states are highly robust to internal stochasticity due to redundancy of critical biological functions. Redundant alternative biological pathways to control the expression of genes/proteins is nature's solution to the short term stochastic behavior of sub-sections of the pathway and are known to exist in abundance in any biological system. To associate high confidence in a GRN, it is imperative that the robustness of cellular steady states is reflected by the robustness of attractors under the stochastic simulations of the corresponding GRNs. Hence, a biologically motivated stochastic model for quantifying the robustness properties of a GRN is essential to compare multiple network configurations for the same biological problem.

In the next section, we formally introduce the SIN and the SIF models of stochasticity and propose algorithms to compute the stochastic cellular differentiation and robustness of attractors in a GRN.

## 6.2 Methods and Techniques

In Chapter 3, we introduced the Boolean function mapping of GRNs. We saw in Section 3.1, how different biological functionalities can be well described using a small set of Boolean functions {BUFF, NOT, AND, OR, IAND}. Truth tables defining the characteristic function of these Boolean functions are shown in Table 6.1. In the presence of stochasticity in Boolean functions, output of these logic gates can be different from those specified by their characteristic functions (represented by  $C^\Delta$  in above truth tables).

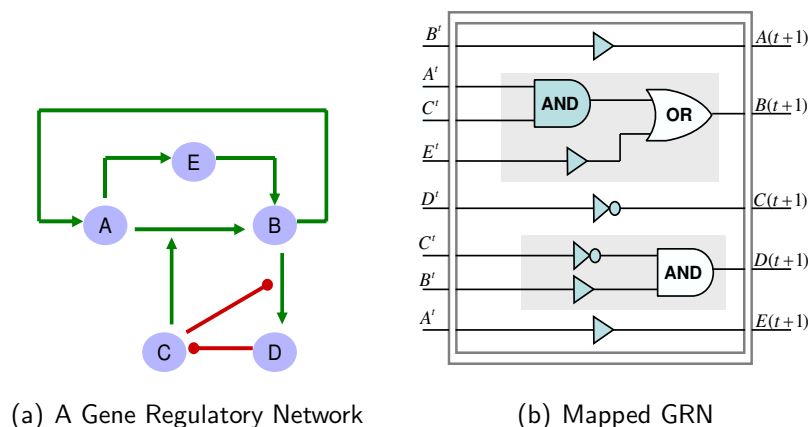
The synthetic GRN from Section 3.1 is reproduced in Figure 6.5 along with the Boolean functions mapped network. As explained earlier in Section 3.1, the shaded Boolean functions in Figure 6.5(b) have a corresponding biological functionality and hence should be modeled for stochastic behavior. How this stochasticity is modeled in a GRN varies in the SIN and SIF models and also depends upon the fault model used (as explained in the next few sections).

### 6.2.1 Fault model

A *fault* in a GRN is defined as the stochastic behavior of a node (i.e. gene expression) or the Boolean function in the GRN. Under the SIN model, faults are modeled in genes expression and under the SIF model, faults are modeled in Boolean functions. Fault in a node or Boolean function  $i$  is represented by a Boolean variable  $\Delta_i$ . If  $\Delta_i = 1$  then gene (or Boolean function)  $i$  takes the faulty value and  $\Delta_i = 0$  represents the normal expression value in the absence of any stochasticity. There are multiple nodes (or Boolean functions) in a GRN and more than one of them can be susceptible to faults. Faults in a GRN at a given instant of time are represented by a Boolean vector  $\Delta$  and is referred to as a *fault configuration* in the GRN.

Here, we make an assumption that at most one gene or one function in a GRN can have a fault at a given instant in time and that multiple faults are spread over different time instants. This implies that  $\Delta_i = 1$  for at most one  $i$  in a fault configuration vector  $\Delta$ . A sequence of network states from a given starting state to an attractor is called the *trajectory* of the state. If  $n$  faults in the network exist then at most  $n$  faults can lie on any trajectory. However, multiple faults cannot exist on a trajectory at the same time instant. We refer to this fault injection model as the *single fault model*. Further, under the single-fault model, given a state of the network, all the possible single faults are independent of each other and can exist with equal probability. This leads to multiple outgoing trajectories from a single state. The assumption of a single fault at a time has been widely used in the literature for stochastic Boolean modeling of GRNs under the SIN model [90, 14, 39, 114]. The single-fault model corresponds to a small probability of two distinct biological functions behaving stochastically at the same instant of time.

Next we modify the transition functions introduced in Chapter 3, for modeling the stochasticity in GRNs.



**Figure 6.5:** (a) A Gene Regulatory Network (GRN). (b) The GRN mapped to Boolean functions (gates).

### 6.2.2 Stochasticity in nodes

In the SIN model, any node can flip its expression, from 0-to-1 or vice versa, due to internal stochasticity in gene regulation mechanisms. When a node flips from its normal expression value due to stochasticity, a fault is said to be injected in the GRN. Fault in the node  $i$  in the GRN is represented by a Boolean variable  $\Delta_i$ . The transition function for a single gene,  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  (Equation 3.3) introduced in Section 3.2 can be modified such that  $x_i^{t+1}$  is equal to the value of the function  $x_i(t+1)$  if there is no fault (i.e.,  $\Delta_i = 0$ ). Otherwise  $x_i^{t+1}$  takes the value opposite to current value of the function (i.e.,  $\neg x_i(t+1)$ ). Equation 6.1 represents the modified transition function  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta)$  in the presence of a fault in the SIN model. Transition function  $T(\mathbf{x}^t, \mathbf{x}^{t+1})$  of the GRN is the same as in Equation 3.4 and is given again here in Equation 6.2.

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) = \left[ (x_i^{t+1} \leftrightarrow x_i(t+1)) \wedge \neg \Delta_i \right] \vee \left[ (x_i^{t+1} \leftrightarrow \neg x_i(t+1)) \wedge \Delta_i \right] \quad (6.1)$$

$$T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) = T_0(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \wedge \dots \wedge T_N(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \quad (6.2)$$

Since any node can flip its expression in the SIN model, there are exactly  $N$  possible faults in the network at a given instance of time, where  $N$  is the number of genes in the network. Under the single-fault model, if the faults in the network are represented by a Boolean vector  $\Delta$  of size  $N$ , at most one gene  $x_i$  has a fault (i.e.,  $\Delta_i = 1$  for at most one bit). Since all the faults are independent of each other, given the state of the network  $\mathbf{x}^t$ , a set of independent and equiprobable fault configuration vector  $\Delta$ s can exist in the network. That is, if we represent the set of possible fault configuration vectors by a set  $D = \{\Delta^1, \Delta^2, \dots, \Delta^N\}$ , the probability of selecting the fault vector  $\Delta^i$  is given by Equation 6.3.

$$P(\Delta = \Delta^i) = \frac{1}{N} \quad (6.3)$$

If the probability of flipping a node  $i$  is given by  $\epsilon^i$ , then the probability that the gene  $i$  has a fault (i.e.,  $P(\Delta_i = 1)$ ) in the fault vector  $\Delta$  is given by Equation 6.4.

$$P(\Delta_i = 1) = \Delta_i \cdot \epsilon^i \quad (6.4)$$

### 6.2.3 Stochasticity in functions

SIF models the stochasticity in biological functions that are represented using Boolean gates AND, OR, BUFF, NOT and IAND in Figure 6.5(b). In the presence of faults, the output of these logic gates can be different from their normal value. For example, the noisy output value of these Boolean functions is given in the last column of Tables 6.1(a)-6.1(e). The noisy output of these Boolean

A	C	C $\Delta$	A	C	C $\Delta$	A	B	C	C $\Delta$	A	B	C	C $\Delta$	A	B	C	C $\Delta$
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	1	1	0	0	1	0	1	1	1	1	0	0

(a) BUFF      (b) NOT      (c) AND      (d) OR      (e) IAND

**Table 6.2:** Truth tables representing the transfer function of different Boolean logic gates.  $A$  and  $B$  are the input genes,  $C$  represents the output gene expression in the absence of stochasticity and  $C^\Delta$  represents the output gene expression in the presence of stochasticity under the SIF model.

functions can take different values depending upon how the faults are modeled. For example, Table 6.2 shows another set of transition functions in the presence of noise. Since, the above Boolean functions represent an underlying biological functionality, it is important that the way faults are modeled in these functions reflects the biological reasoning behind stochasticity in underlying gene or protein regulation mechanism. Keeping this in mind, Table 6.2 has a closer correspondence with biological mechanisms than Table 6.1 as explained below.

In Tables 6.2(a)-6.2(e), noise has an impact on the function only when all the positive inputs are “active” or 1. This constraint corresponds to the fact that a biological function can behave stochastically only when it is functionally active. For example, transcription of a gene can take place only when the transcription factor is present and there is a natural stochasticity involved in the process of transcription. On the other hand, if the transcription factor is absent, there can be no stochasticity in the transcription process and the gene would never be expressed. Boolean gates BUFF, NOT and AND have all the input ports as positive inputs. Boolean gate IAND have some ports which go through the NOT gate and act as negative inputs. Boolean OR gate is modeled to have no stochasticity because it just represents that the two alternate biological functions can have an impact on the same gene/protein. Note that the noise in these alternate biological functions is already modeled with the remaining stochastic gates (i.e., AND, NOT and BUFF).

In the SIN model, the fault variable  $\Delta$  represents stochasticity in the expression of a gene whereas in the SIF model,  $\Delta$  represents stochasticity in Boolean functions. Since not all Boolean functions in a GRN behave stochastically, let us define  $G = \{G_1, G_2, \dots, G_p\}$  as a set of stochastic functions in the mapped GRN of Figure 6.5(b). If  $\Delta_i = 1$ , then Boolean function  $G_i$  behaves stochastically and take the expression value as defined by the last column of truth tables 6.2(a)-6.2(e). Otherwise, if  $\Delta_i = 0$ , Boolean function  $G_i$  behaves per its original description. Equations 6.5-6.9 formally describe the stochastic Boolean functions.

$$\text{BUFF} : f^B(x_a) = [(x_c \leftrightarrow \mathbf{0}) \wedge \Delta] \vee [(x_c \leftrightarrow x_a) \wedge \neg\Delta] \quad (6.5)$$

$$\text{NOT} : f^N(x_a) = [(x_c \leftrightarrow \mathbf{1}) \wedge \Delta] \vee [(x_c \leftrightarrow \neg x_a) \wedge \neg \Delta] \quad (6.6)$$

$$\text{OR} : f^O(x_1, \dots, x_p) = (x_c \leftrightarrow \bigvee_{i=1}^p x_i) \quad (6.7)$$

$$\text{AND} : f^A(x_1, \dots, x_p) = [(x_c \leftrightarrow \mathbf{0}) \wedge \Delta] \vee \left[ (x_c \leftrightarrow \bigwedge_{i=1}^p x_i) \wedge \neg \Delta \right] \quad (6.8)$$

$$\text{IAND} : f^{IA}(x_1, \dots, x_p) = \left[ \{x_c \leftrightarrow (\bigwedge_{i=1}^{p^{in}} \neg x_i^{in} \wedge \bigwedge_{j=1}^{p^a} x_j^a)\} \wedge \neg \Delta \right] \vee \left[ \{x_c \leftrightarrow \bigwedge_{j=1}^{p^a + p^{in}} x_j\} \wedge \Delta \right] \quad (6.9)$$

The expression of each gene  $i$  at time  $t + 1$  is specified by the function  $x_i(t + 1)$  of the state of the genes acting as its input at time  $t$ . The construction of function  $x_i(t + 1)$  was given in Equation 3.1 (Chapter 3). Alternatively, the function  $x_i(t + 1)$  can be formed by composing Boolean gates as in Figure 6.5(b) and using the corresponding Equations 6.5-6.9.

**Example 6.2.1** For the node B in the GRN in Figure 6.5,  $x_B(t + 1)$  is defined in Equation 6.10.

$$x_B(t + 1) = f^O(f^B(x_E^t), f^A(x_A^t, x_C^t)) \quad (6.10)$$

In the above equation,  $f^B$ ,  $f^A$  and  $f^O$  are defined in Equations 6.5, 6.8 and 6.7 respectively, and  $x_C^t$  and  $x_E^t$  represent the expression of nodes  $C$  and  $E$  at the time instant  $t$ . ■

The transition function  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$  of a gene  $i$  for SIF model is given by Equations 6.11-6.12 for the deterministic Boolean networks. Note that whereas the SIN model modifies the transition function of a gene, the description of Boolean functions is modified in the SIF model.

$$T_i(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) = \{x_i^{t+1} \leftrightarrow x_i(t + 1, \Delta)\} \quad (6.11)$$

$$T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) = T_0(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \wedge \dots \wedge T_N(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \quad (6.12)$$

Given a state of the network  $\mathbf{x}^t$ , not all Boolean functions in the set  $G$  behave stochastically. We use a Boolean vector  $\Delta$  of size  $|G|$  to represent the faulty Boolean functions in the network. In a given fault vector  $\Delta$ , the bit  $\Delta_i = 1$  only if all the positive inputs to the function  $G_i$  are active or 1. Hence, the number of faults in the network at a given time instant  $t$  depends upon the current state of the network. Again, assuming the single-fault model, a set  $D = \{\Delta^1, \Delta^2, \dots, \Delta^{|D|}\}$  of independent fault vectors  $\Delta^i$  may exist such that in each fault vector, at most one Boolean function  $G_i$  has a fault (i.e.,  $\Delta_i = 1$

for at most one bit). The probability of selecting the fault vector  $\Delta^i$  is given by Equation 6.13.

$$P(\Delta = \Delta^i) = \frac{1}{|D|} \quad (6.13)$$

Boolean functions in the set  $G$  correspond to Biological functions and have a probability of failure  $\epsilon^i$  associated with each  $G_i$ . The probability of failure  $\epsilon^i$  is independent of the state of the network and solely depends upon the complexity of the biological function that it represents. The probability that Boolean function  $i$  has a fault (i.e.,  $P(\Delta_i = 1)$ ) in a given fault vector is given by Equation 6.14.

$$P(\Delta_i = 1) = \Delta_i \cdot \epsilon^i \quad (6.14)$$

Whereas the set  $D$  of fault configuration vectors does not depend upon the state of the network and is always the same in the SIN model, both the size of the set  $D$  and its elements depend upon the current state of the network  $\mathbf{x}^t$  in the SIF model. Further, the size of fault configuration vector  $\Delta$  is different for the SIN and the SIF models. Whereas the size of the vector  $\Delta$  is equal to the number of genes in the network in the SIN model, it is equal to the number of stochastic gates in the SIF model of stochasticity.

With these two models of stochasticity in GRNs, we provide, in the next two sections, algorithms to compute the probability of cellular differentiation and robustness.

## 6.2.4 Algorithms for stochastic cellular differentiation

Given a state of the network  $\mathbf{x}^t$ , a fault vector  $\Delta$  and the transition function  $T_i(\mathbf{x}^t, \mathbf{x}^{t+1})$ , the state of the network in the next time step,  $\mathbf{x}^{t+1}$ , can be computed by using Equation 6.15 where the symbol  $\exists$  stands for existential quantification.

$$\mathbf{x}^{t+1} = \exists_{\Delta} \exists_{x^{t+1}} \{T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \wedge \mathbf{x}^t \wedge \Delta\} \quad (6.15)$$

Given a fault vector  $\Delta$  of length  $n$  and a current state of the network  $\mathbf{x}^t$ , the probability that the network would exist in the faulty state  $\mathbf{x}^{t+1, \Delta}$  and the fault-free state  $\mathbf{x}^{t+1}$  is given by Equations 6.16 and 6.17 respectively.

$$P(\mathbf{x}^{t+1, \Delta} | (\mathbf{x}^t, \Delta)) = \sum_{i=1}^n P(\Delta_i = 1) \quad (6.16)$$

$$P(\mathbf{x}^{t+1} | (\mathbf{x}^t, \Delta)) = \sum_{i=1}^n (1 - P(\Delta_i = 1)) \quad (6.17)$$

By applying Bayes' rule on Equations 6.16 and 6.17, the probability of the network being in state  $\mathbf{x}^{t+1}$  at the next time instant is given by Equations 6.18-6.22. In Equation 6.18, we marginalise the probability over all possible fault



configuration vector  $\Delta$ s. By using Equation 6.13 in Equation 6.18, we get the probability of generating the faulty next state  $\mathbf{x}^{t+1,\Delta}$  from the current state  $\mathbf{x}^t$  in Equation 6.19. If the network can be in only one starting state  $\mathbf{x}^t$ , the probability of generating the faulty state  $\mathbf{x}^{t+1,\Delta}$  is given by Equation 6.20.

$$P(\mathbf{x}^{t+1,\Delta}|\mathbf{x}^t) = \sum_{\delta \in D} \{P(\mathbf{x}^{t+1,\Delta} | (\mathbf{x}^t, \Delta = \delta)) \cdot P(\Delta = \delta)\} \quad (6.18)$$

$$= \frac{1}{|D|} \sum_{\delta \in D} P(\mathbf{x}^{t+1,\Delta} | (\mathbf{x}^t, \Delta = \delta)) \quad (6.19)$$

$$P(\mathbf{x}^{t+1,\Delta}) = P(\mathbf{x}^{t+1,\Delta} | \mathbf{x}^t) \cdot P(\mathbf{x}^t) \quad (6.20)$$

A similar set of equations exist for the fault-free next state  $\mathbf{x}_{t+1}$  (Equations 6.21 and 6.22).

$$P(\mathbf{x}^{t+1}|\mathbf{x}^t) = \frac{1}{|D|} \sum_{\delta \in D} P(\mathbf{x}^{t+1} | (\mathbf{x}^t, \Delta = \delta)) \quad (6.21)$$

$$P(\mathbf{x}^{t+1}) = P(\mathbf{x}^{t+1} | \mathbf{x}^t) \cdot P(\mathbf{x}^t) \quad (6.22)$$

If the network may exist in a set of initial states  $S$  and the probability of each initial state is specified, then the probability of being in the next states

---

**Algorithm 11:** Algorithm for computing probability of faulty next states.

---

```

1 stochastic_next_states( $T, S_t, P_t, G$ )
2 begin
3    $S_{t+1} = \emptyset$ 
4   for  $i = 0$  to  $|S_t|$  do
5      $D = \text{construct\_fault\_config}(S_t^i, G)$ 
6      $\Delta = \mathbf{0}$ 
7      $s_{tmp} = \exists_{\Delta} \exists_{x^{t+1}} \{T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \wedge S_t^i \wedge \Delta\}$ 
8     for  $j = 0$  to  $|D|$  do
9        $s_{tmp}^{\Delta} = \exists_{\Delta} \exists_{x^{t+1}} \{T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta) \wedge S_t^i \wedge D^j\}$ 
10       $P(s_{tmp}^{\Delta}) = P(s_{tmp}^{\Delta} | S_t^i) \cdot P_t^{S_t^i}$ 
11       $P(s_{tmp}) = P(s_{tmp} | S_t^i) \cdot P_t^{S_t^i}$ 
12       $P_{t+1}^{\Delta} = P_{t+1}^{s_{tmp}^{\Delta}} + P(s_{tmp}^{\Delta})$ 
13       $S_{t+1} = S_{t+1} \cup s_{tmp}^{\Delta}$ 
14       $P_{t+1}^{s_{tmp}} = P_{t+1}^{s_{tmp}^{\Delta}} + P(s_{tmp})$ 
15       $S_{t+1} = S_{t+1} \cup s_{tmp}$ 
16   return  $(S_{t+1}, P_{t+1})$ 
17 end
```

---

---

**Algorithm 12:** Algorithm for computing probability of differentiation into various attractors in the presence of up to  $k$  faults.

---

```

1 stochastic_differentiation_k_faults( $T, S, G, k, SS$ )
2 begin
3   for  $i = 1$  to  $|S|$  do
4      $P_t^{S^i} = 1/|S.size()$ 
5   for  $i = 1$  to  $|SS|$  do
6      $P_{SS_i} = 0$ 
7    $S_t = S$ 
8   for  $i = 1$  to  $k$  do
9      $(S_{t+1}, P_{t+1}, \tilde{P}_{SS}) = \text{stochastic\_differentiation}(T, S_t, G, P_t, SS)$ 
10    for  $j = 1$  to  $|SS|$  do
11       $P_{SS_j} = P_{SS_j} + \tilde{P}_{SS_j}/k$ 
12     $t = t + 1$ 
13  return  $P_{SS}$ 
14 end

15 stochastic_differentiation( $T, S, G, P, SS$ )
16 begin
17    $(S_{t+1}, P_{t+1}) = \text{stochastic\_next\_states}(T, S, P, G)$ 
18   for  $j = 1$  to  $|SS|$  do
19     for  $i = 1$  to  $|S_{t+1}|$  do
20       if  $BR(SS_j) \cap S_{t+1}^i \neq \emptyset$  then
21          $P_{SS_j} = P_{SS_j} + P_{t+1}^{S_{t+1}^i}$ 
22   return  $(S_{t+1}, P_{t+1}, P_{SS})$ 
23 end

```

---

$\mathbf{x}^{t+1, \Delta}$  and  $\mathbf{x}^{t+1}$  is given by Equations 6.23 and 6.24, respectively.

$$P(\mathbf{x}^{t+1, \Delta}) = \sum_{x_t \in S} P(\mathbf{x}^{t+1, \Delta} | \mathbf{x}^t) \cdot P(\mathbf{x}^t) \quad (6.23)$$

$$P(\mathbf{x}^{t+1}) = \sum_{x_t \in S} P(\mathbf{x}^{t+1} | \mathbf{x}^t) \cdot P(\mathbf{x}^t) \quad (6.24)$$

Algorithm 11 describes how the probability of the set of next states  $S_{t+1}$  is computed from a given set of initial states  $S_t$ . In line 5 of Algorithm 11, the possible fault configuration vectors are computed from an initial state  $S_t^i$  ( $i = 1, 2, \dots, |S_t|$ ). For each fault configuration vector in the set  $D$ , the next states are generated in line 9 and the probability of each next state is computed in lines 10-14 by using Equations 6.18-6.24. In Algorithm 11,  $P_t$  and  $P_{t+1}$  represents the probability of states in the set  $S_t$  and  $S_{t+1}$  respectively.

Algorithm 12 describes how the probability of transition into different steady states can be computed from a given set of initial states. In Algorithm 12, given a set of states  $S$ , backward reachable set  $BR(S)$  is a set of all the states of the network that can make a transition into the states in  $S$  in one or more time steps under the no-stochasticity condition (i.e.,  $\Delta = \mathbf{0}$ ).

---

**Algorithm 13:** Algorithm for computing Robustness of Attractors in the presence of up to  $k$  faults.

---

```

1 robust_attractors_k_faults( $T, S, G, k$ )
2 begin
3    $\tilde{T} = \exists_{\Delta} [\{T(\mathbf{x}^t, \mathbf{x}^{t+1}, \Delta)\} \wedge \{\Delta = \mathbf{0}\}]$ 
4    $SS = \text{all\_attractors}(\tilde{T})$ 
5   for  $i = 1$  to  $|SS|$  do
6      $P_{SS}[i] = \text{stochastic\_differentiation\_k\_faults}(T, SS_i, G, k, SS)$ 
7 end
```

---

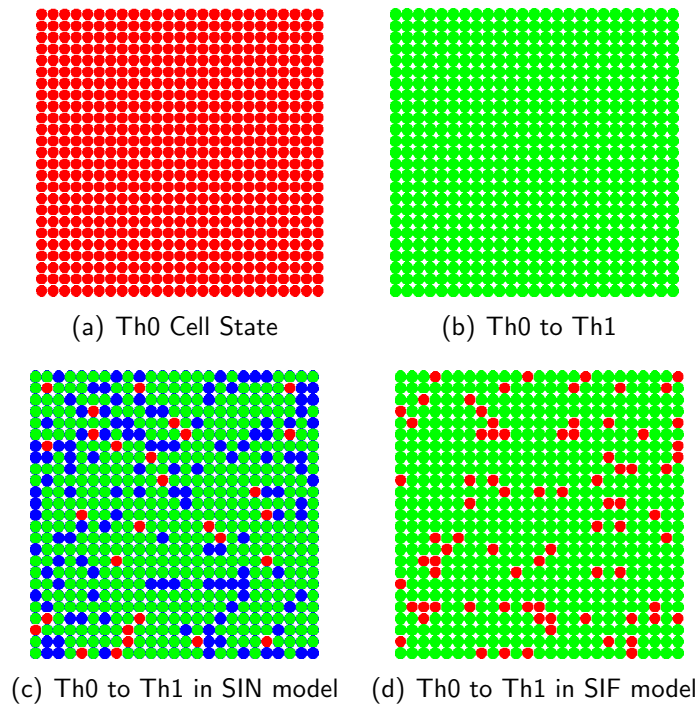
If  $SS_{a_i}$  represents the set of states in an attractor  $a_i$ , one can test if the current state of the network  $\mathbf{x}^t$  can differentiate into the attractor  $a_i$  given a fault vector  $\Delta$  by testing if  $BR(SS_{a_i}) \cap \mathbf{x}^{t+1} \neq \emptyset$ , where  $\mathbf{x}^{t+1}$  is computed using Equation 6.15. The probability of making a transition to an attractor  $a_i$  is then given by the sum of  $P(\mathbf{x}^{t+1})$  for all the states  $\mathbf{x}_{t+1}$  that can make a transition to  $a_i$ . The function `stochastic_differentiation()` in lines 15-23 of Algorithm 12 computes the probability of differentiation into all the cellular steady states from an initial set of states  $S$ . In line 17, we compute the probability of all the faulty states that may exist by injecting a single fault in the network. Multiple faults may exist in the network that we model using the function `stochastic_differentiation_k_faults()` in lines 1-14. This function models  $k$  sequential faults in the network. However, each fault is injected under the single-fault model and multiple faults exist only in consecutive time steps.

### 6.2.5 Algorithm for robustness computation

In the absence of any stochasticity, attractors in a GRN are first computed using the Algorithm 2 proposed in Chapter 3 for the deterministic Boolean modeling of GRNs. By the definition of an attractor (see Section 3.2.3), there can be no path among the attractors in a deterministic model. If  $SS_{a_i}$  represents the states in the attractor  $a_i$ , then by using the function `stochastic_differentiation_k_faults()`, defined in Algorithm 12, the probability of differentiation into various attractors can be computed for every attractor  $a_i$ . The probability to differentiate into various attractors in turn represents the robustness of an attractor  $a_i$ . Algorithm 13 formally describes the procedure to compute the robustness of attractors.

## 6.3 Simulation results

In this section, the SIN and SIF models of stochasticity are applied on two real GRNs proposed in the literature, namely: T-Helper network [102] and T-Cell activation network [140]. These GRNs are modeled under increasing number of faults in the network. The results and discussion is organized under the



**Figure 6.6:** Simulation results showing the effect of noise on T-helper cell differentiation process with an external stimulus of  $IFN\gamma$ . Each small circle is representative of a T-Helper cell and each cell is modeled to behave independent of the neighbouring cells. Red cells represent the naïve undifferentiated Th0 cells, green cells represent Th1 cell state and blue cells represent Th2 cell state. Ratio of number of red (green or blue) cells to total number of cells in a panel is representative of the probability of differentiating into Th0 (Th1 or Th2) cell state. (a) Cell Culture maintained in Th0 state. (b) In absence of any stochasticity all Th0 cells differentiate to Th1 cell state on receiving  $IFN\gamma$ . (c) Th0 cells differentiate into Th1 and Th2 under the SIN model of stochasticity. Few cells revert to Th0 state as seen by the few patches of red color. (d) SIF model of stochasticity shows that Th0 cells differentiate into Th1 cells while some cells cannot differentiate on receiving  $IFN\gamma$  and revert to Th0 cell state. None of the cells differentiate into Th2 cell state. The probability of failure (i.e.,  $\epsilon_i$ ) is 0.5 for all the nodes (functions) in the SIN model (SIF model). The stochasticity models are simulated with upto 4 faults in a GRN.

earlier mentioned two properties of steady states, i.e., cellular differentiation and robustness of attractors.

### 6.3.1 Cellular differentiation

#### T-Helper network

On simulating the Th0 to Th1 cellular differentiation in response to external  $IFN\gamma$  stimulus under the SIN model of stochasticity, we found that an almost equal number of cells differentiate into Th1 and Th2 from the Th0 cell state

and a few cells revert to Th0 (Figure 6.6(c)). Biologically it is known that Th0 cells cannot differentiate to Th2 state in response to an  $\text{IFN}\gamma$  stimuli [94]. The difference in the simulation results from the known biological observation can be a shortcoming of the GRN, of Boolean modeling, or of the model of stochasticity. The equally likely cellular differentiation of steady states under the SIN model of stochasticity has been observed earlier in [98] and was tagged as a shortcoming of Boolean models. However, in our opinion, SIN model of stochasticity is the main reason behind this discrepancy in simulation results. If we use the more biologically motivated SIF model of stochasticity, where the stochasticity in a biological function is tightly linked to activity of other nodes in the network, we see that a major sub-population of Th0 cells differentiate into Th1 cellular state and a few cells revert to Th0 in response to  $\text{IFN}\gamma$  dosage (Figure 6.6(d)). This is consistent with the expected biological behavior of T-Helper cells [94] and thereby makes a strong case for the refined SIF model.

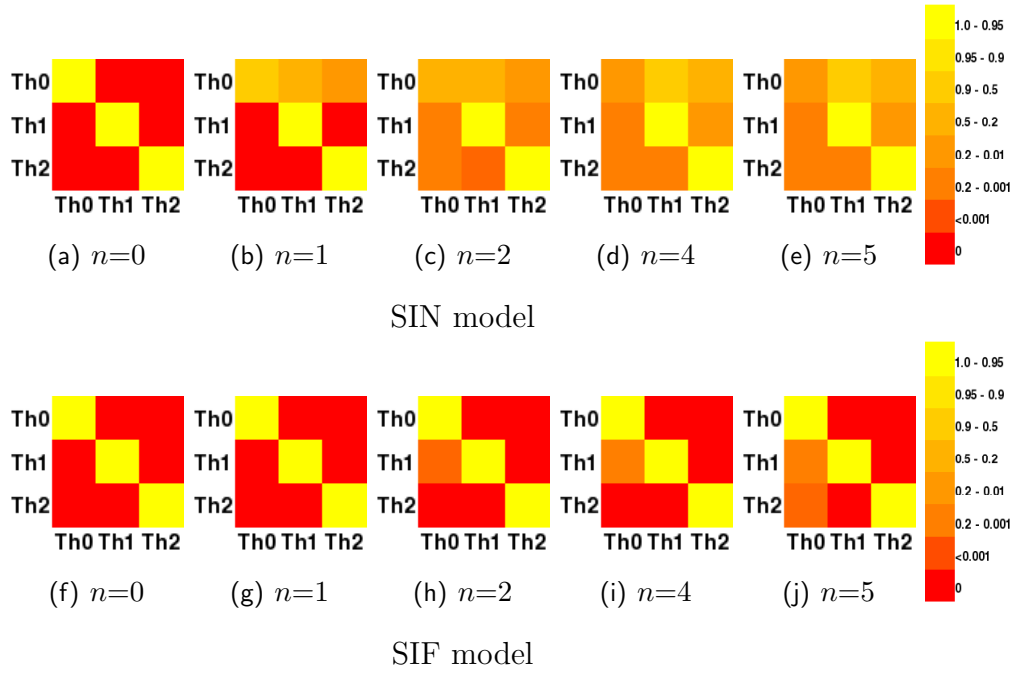
### T-Cell activation Network

Unlike the T-Helper network, the T-Cell activation network does not differentiate into different cell types. Attractors of the T-Cell GRN just represent the gene expression profiles of differentially activated T-Cells. Hence, the cellular differentiation property is not applicable to the study of the T-Cell activation GRN.

## 6.3.2 Robustness of attractors

### T-Helper network

Robustness results of the T-Helper network under the SIN and the SIF models of stochasticity are shown in Figure 6.7. Since, in the absence of any stochasticity (i.e.  $n = 0$ ), an attractor cannot make a transition to another attractor, Figures 6.7(a) and 6.7(f) have non-red entries only along the diagonal. In Figure 6.7, under the SIN model, all the three attractors (i.e., Th0, Th1 and Th2) are found to make a transition into each other with a significant probability (represented by the intensity of the yellow color in Figure 6.7). Robustness of attractors is measured as the number of faults in the network are increased from 0 (i.e., no stochasticity) to 5 faults. One can see from the top row in Figure 6.7 that robustness decreases if the number of faults in the network is increased under the SIN model. This observation could be specific to the GRN of T-Helper network but a similar observation can be made on the T-Cell activation network. As any GRN is not robust under the SIN model, it may not be a good model of stochasticity for comparing different GRNs. Under the SIF model, Th0 cell state is found to be robust to stochasticity (bottom row of Figure 6.7). Th1 and Th2 cellular states are very robust as most of the cells stay in the original attractor state even with 5 sequential faults in the network. Moreover, Th1 and Th2 cells do not show transition among each

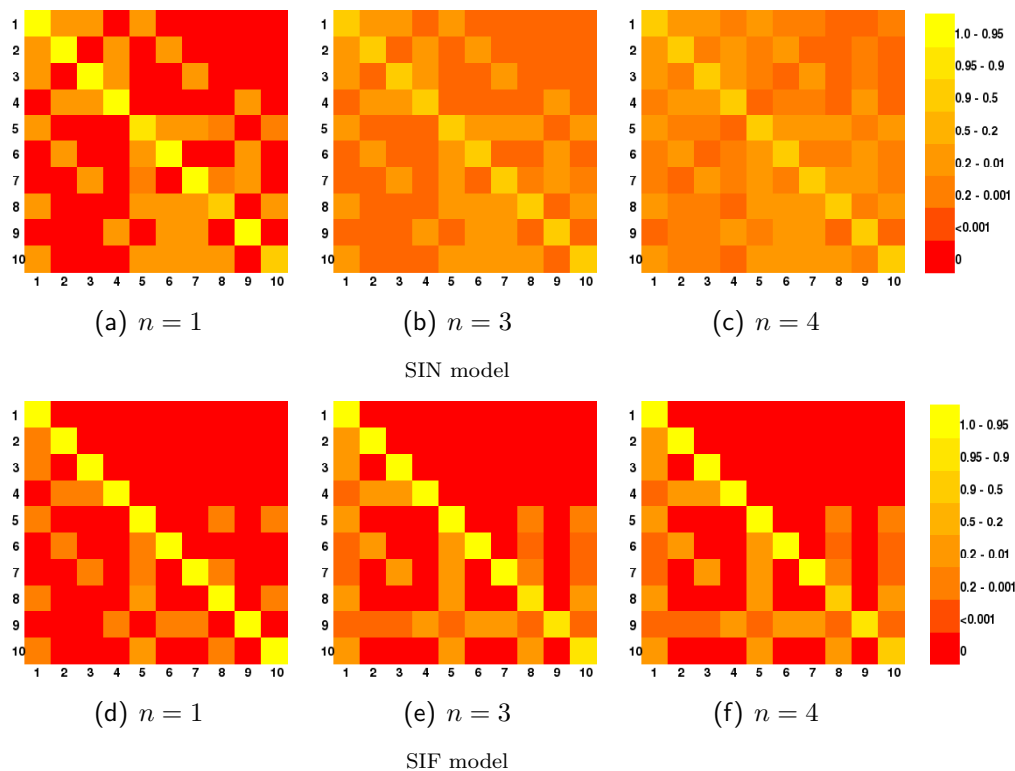


**Figure 6.7:** Simulation results showing the transition probability among Th0, Th1 and Th2 cell states of the T-Helper network. (a)-(e) Transition probabilities in the SIN model as the number of faults  $n$  in the network is increased from  $n = 0$  to  $n = 5$ . (f)-(k) Transition probabilities in the SIF model. In each figure, the intensity of yellow color in the entry  $i$ - $j$  corresponds to the probability of transition from the attractor  $i$  to the attractor  $j$ . The colorbar in the rightmost column indicates the color-probability encoding. The probability of failure (i.e.,  $\epsilon_i$ ) is 0.5 for all the nodes (functions) in the SIN model (SIF model).

other as the number of faults is increased. This observation further increases our confidence in the SIF model as biologically Th0, Th1 and Th2 cell states are known to be robust and the underlying T-Helper network of Figure 6.3 is a well established GRN in the literature.

### T-Cell activation Network

We next applied the SIN and SIF models of stochasticity on T-Cell activation network from [140]. We measured the probability of reachability among the attractors with an increasing number of faults in the network. The T-Cell activation network has 10 attractors in the absence of stochasticity. As the number of faults is increased, the number of red entries in the heatmaps of Figure 6.8 decreases showing the decreasing robustness of different attractors. The intensity of yellow color in a cell is proportional to the probability of transition among the corresponding attractors labeled on the X and Y axes. For single fault injection, one can already see that the probability of transitions among these attractors in the SIN model is more widespread than in SIF model on the same network. To test if SIN and SIF models show the same reachability



**Figure 6.8:** Simulation results showing the transition probability among the ten attractors of the T-Cell activation network. (a)-(d) Transition probabilities in the SIN model as the number of faults  $n$  in the network is increased from  $n = 0$  to  $n = 4$ . (e)-(h) Transition probabilities in the SIF model. The colorbar in the rightmost column indicates the color-probability encoding. The probability of failure (i.e.,  $\epsilon_i$ ) is 0.5 for all the nodes (functions) in the SIN model (SIF model).

among the attractors with an increasing number of faults, we simulated the injection of 4 faults sequentially. Just after three faults, almost all the attractors show transitions among each other in the SIN model (Figure 6.8(b)). The results are similar to the those seen earlier for the T-Helper network, where Th0, Th1 and Th2 cell states can transition among each other in response to internal stochasticity (Figure 6.7). In Figure 6.8, under the SIF model, transitions among the attractors is sparse and a saturation in transition probabilities is observed as the number of faults in the network increases. Since the SIF model is closer to the biological phenomenon of inducing faults in biological functions and does not always give low robustness measure, it can provide an effective way to compare the robustness of two different configurations of GRNs in response to internal stochasticity.

From the results above it is evident that the SIN model of stochasticity often leads to over-representation of noise in GRNs by making all the genes/proteins equally likely to flip, independent of the expression of the input genes and complexity of the underlying biological function. With the improved SIF model of stochasticity, it is possible to simulate biological phenomena such

as gene perturbation experiments more accurately and to construct GRNs that exhibit strong robustness properties.

## 6.4 Summary

In this chapter, we have extended the deterministic Boolean formalism proposed in Chapter 3 to stochastic Boolean modeling of GRNs. A new method, called *stochasticity in functions* (SIF), has been proposed for modeling the stochasticity in GRNs. Unlike the traditional *stochasticity in nodes* (SIN) model that simulates stochasticity by flipping gene expression values, SIF models the stochasticity induced at the level of biological functions. SIF associates a probability of failure with different biological functions and models stochasticity in these functions depending upon the expression of the input nodes. By applying the SIN and SIF models on the T-Helper network, we have shown that while the SIN model predicts biologically implausible behavior of T-Helper cell differentiation, the SIF model correctly predicts the Th0 to Th1 cellular differentiation process. Further, by analysing the robustness of steady states of the T-Helper and T-Cell activation networks it was shown that the SIN model predicts low robustness properties in both cases whereas the SIF model predicts more biologically relevant behavior with higher robustness.



---

# Modeling Cell Growth vs. Apoptosis

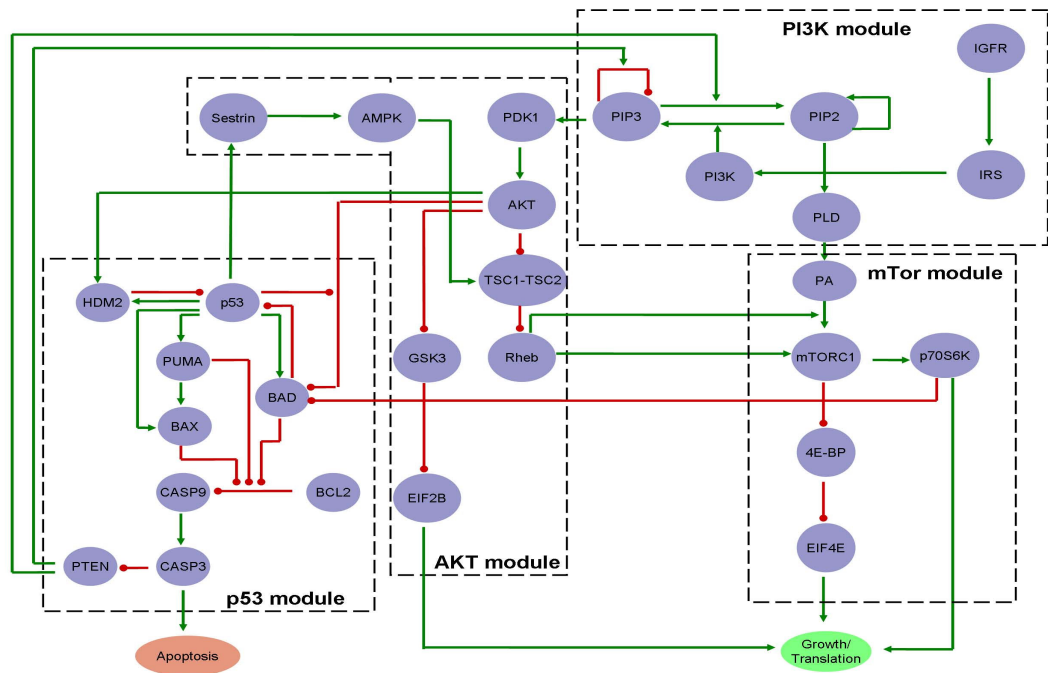
---

# 7

During its lifetime, a cell grows in response to environmental nutrients, undergoes proliferation and apoptosis. Cellular apoptosis can be either due to either an unexpected malfunctioning in the cell machinery or as a result of natural aging process. In a normal healthy tissue, a balance is always maintained between the number of cells undergoing apoptosis and cell growth. At the single cell level, this balance is the result of strict self-regulation of pro-apoptotic and pro-growth signals generated by different proteins inside the cell. The presence of feedback loops among the proteins inside a cell ensures that cells neither undergo an uncontrolled growth (leading to formation of tumors) nor they have too much of apoptosis (which may lead to diseases such as Alzheimer's and Parkinson's).

Some cells may carry mutations in a few genes that are functional in the feedback loops which maintain a proper balance between generating pro-growth and pro-apoptotic signals inside a cell. If these mutations are in the favour of pro-growth signals, then a cell is said to have a pre-disposition towards uncontrolled growth (and hence proliferation). In such a scenario, a cell (carrying mutations) undergoes uncontrolled proliferation and consequently leads to the formation of tumors. Most cancer treatments either try to restore the normal expression of mutated genes directly or counteract the impact of mutated genes by targeting other genes (or proteins) in the pathway. Understanding how different genes (and proteins) regulate each other in cancer cells is therefore of major interest in the development of treatments for this disease.

In this chapter, we illustrate an application of algorithms, proposed in the previous chapters, on the GRN that models a balance between the “apoptosis” and the “growth” signals in a healthy cell. In Section 7.1, we start with the description of a GRN that models the balance between apoptosis and growth. The explanation of the GRN follows the sequence of steps followed while con-



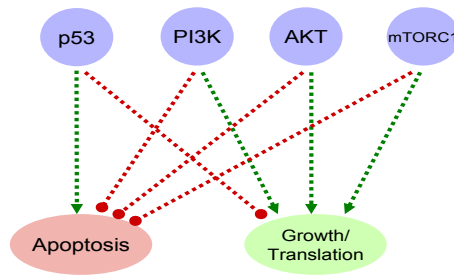
**Figure 7.1:** GRN representing interactions among some proteins known to play a crucial role in maintaining a balance between apoptosis and cellular growth.

structuring the pathway. Starting from the key proteins PI3K, mTOR, AKT and p53 we construct the GRN incrementally by including proteins known to interact with these base proteins in the literature. Finally, in Section 7.2 we show simulation results of modeling a few known gene mutations that have been implicated in the past for uncontrolled growth or for abnormally low apoptosis of mutated cells.

## 7.1 Cell survival vs. apoptosis GRN

Some of the key proteins that are known to play a crucial role in the normal functioning of a cell are summarised in the GRN of Figure 7.1. Interactions proposed in the GRN of Figure 7.1 have been collected from the published literature on the experimental study of these proteins to model the cancer behavior of a cell. As one can see from Figure 7.1, by just looking at the pathway it becomes difficult to understand how different proteins work in tandem with each other and influence the cell growth and apoptosis signals.

The GRN in Figure 7.1 has been constructed in a bottom-up strategy starting with an abstracted pathway comprising of the four proteins PI3K, mTOR, AKT and p53 (Figure 7.2). PI3K, mTOR and AKT are known to participate in the anti-apoptotic, pro-survival pathways in the cells. On the other hand, p53 is known to participate in generating pro-apoptotic and anti-survival signals. Starting from this knowledge, we first expand upon the p53 module of



**Figure 7.2:** Topmost abstraction level of the key players in cell growth and apoptosis pathways.

the GRN and include proteins downstream of p53 that have been implicated for the cellular apoptosis in the literature. Then we expand upon the proteins known to participate in AKT, PI3K and mTOR pathways respectively.

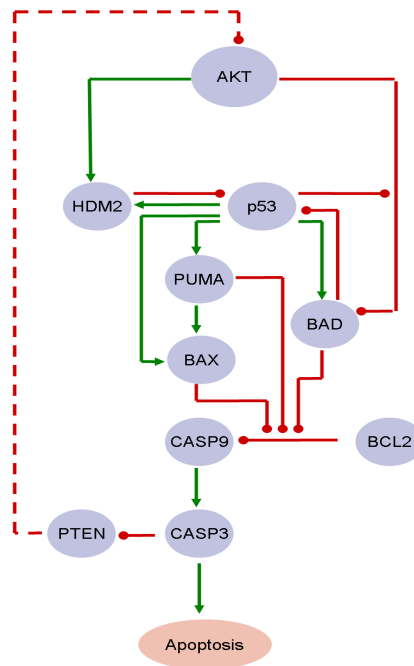
The four modules of the pathway crosstalk through various intermediate proteins. Therefore, while describing a specific module, we will only focus on proteins that are directly downstream or upstream of the key regulator of the module (i.e. PI3K for PI3K module, p53 for p53 module, etc) and use dashed edges to abstract away the information on a few protein interactions which are further elaborated while describing other modules.

### 7.1.1 P53 module

The p53 protein, as we have explained before, plays a crucial role in generating pro-apoptotic and anti-growth signals when a malfunctioning is detected in the cell or cell is exposed to some external stress such as lack of nutrients or oxygen. It exhibits its functionality through various intermediate proteins as sketched in Figure 7.3. The biological explanation behind different interactions in the GRN module of Figure 7.3 is summarised below.

#### AKT-p53-MDM2 interaction

In normal cells, p53 is known to exist in very small concentration inside the nucleus [104]. When p53 is present in the transcriptionally active form, it activates various target genes including MDM2, PUMA and BAD [89, 77, 123, 124]. HDM2 protein, a human analogous of mouse MDM2, acts as a ubiquitin ligase of p53 [73]. The HDM2 protein when activated (on phosphorylation by AKT [103]), trans-locates into the nucleus and binds with the p53 protein. The HDM2 bound p53 is then trans-locates itself out of the nucleus into the cytoplasm where it is degraded [104]. The counter interaction between HDM2 and p53 provides a negative feedback loop that controls the adequate amount of p53 in the normal cells.



**Figure 7.3:** p53 module for modeling the cell apoptosis.

### AKT-p53-BAD interactions

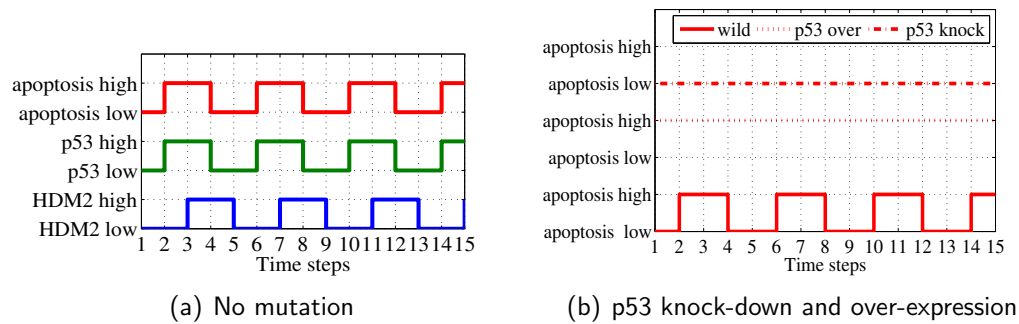
Activated p53 is involved in generating apoptosis signals via proteins PUMA, BCL2, BAX, BAD, Caspase 9 and Caspase 3. p53 can transcriptionally activate PUMA [89, 77] and BAD [123, 124]. BAD undergoes rapid phosphorylation by AKT and normally exists in the cytosol in the hyper-phosphorylated form [146]. BAD in its phosphorylated form loses its pro-apoptotic functionality. In the presence of pro-apoptotic signals (such as high p53 activity), phosphorylated BAD undergoes dephosphorylation and forms a complex with p53 [123, 124]. The resulting complex trans-locates itself to the mitochondria where it exerts its pro-apoptotic activity.

### p53-PUMA-BAX interactions

Proteins p53 and PUMA can conformally change the BAX, which is present in the inactive form in the cytosol in the normal circumstances [112, 84]. The activated conformation, trans-locates BAX to mitochondria where it can perform the pro-apoptotic functions. PUMA can also directly trans-locate to mitochondria where it performs anti-apoptotic functions.

### PUMA-BAD-BAX-Caspase interactions

The BCL2 protein, which is normally present on mitochondria has anti-apoptotic functionality and restricts the *cytochrome c* inside the mitochondria. *Cytochrome c* when released into the cytosol leads to activation of caspase cascade



**Figure 7.4:** Simulation results for p53 module.

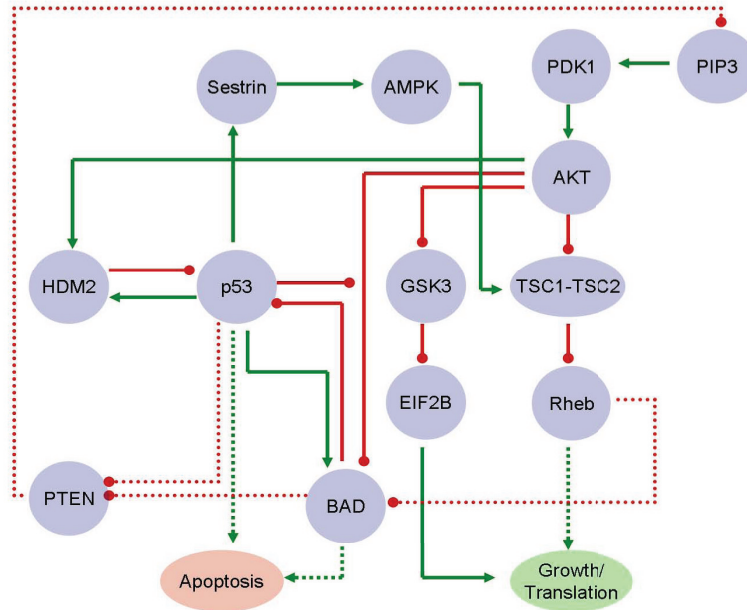
involving sequential activation of CASPASE 9 and CASPASE 3 which acts as an ultimate step before the cell undergoes apoptosis by DNA degradation [60]. BAD, BAX and PUMA dimerize with BCL2 leading to loss of its anti-apoptotic activity [89, 77, 84]. This action further leads to formation of pores in the mitochondria that in turn releases cytochrome c, leading to activation of caspase signaling cascade. This phenomenon is critically dependent upon the balance between the anti-apoptotic and pro-apoptotic signals. Further PTEN, the details of which we will see in Section 7.1.3, undergoes cleavage in the presence of caspase 3 [76]. PTEN is a tumor suppressor gene and is also known to be frequently mutated in human cancers [72, 25, 109].

### p53 module analysis

To ensure that the small module in Figure 7.3 correctly captures the p53 functionality, we simulated the steady state behavior using `genYsis`. All the genes in the p53 module in Figure 7.3 were modeled at the binary activation levels: present or absent. In the absence of any mutation we found one attractor (steady state) with an oscillating “apoptosis” node, exhibiting the balance maintained in generating the pro-apoptotic signals (Figure 7.4). With over-expressed p53 or knock-down p53 (resembling p53 mutations), we get one stable steady state where the apoptosis signal is constitutively 1 or 0 respectively showing the pro-apoptotic functionality of p53.

### 7.1.2 AKT module

The AKT protein (also known as *protein kinase B*, PKB) plays an important role in regulating the cell survival and proliferation. It has also been an important subject of research for cancer therapies as it has been found in its hyper-active form in various cancer cells. AKT acts as both anti-apoptotic and pro-growth through various intermediate proteins as summarised in Figure 7.5.



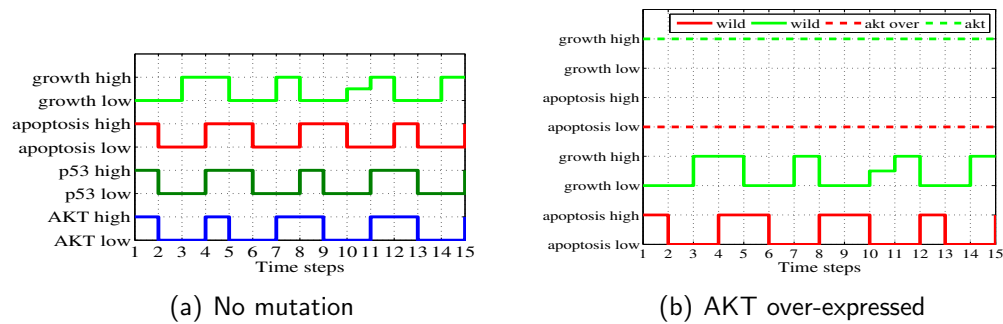
**Figure 7.5:** AKT pathway for modeling the cellular growth and apoptosis.

### PIP3-PDK1-AKT interactions

Once the protein *phosphatidylinositol (4,5)-triphosphate* (PIP3) is activated (details in Section 7.1.3), it recruits *phosphoinositide-dependent protein kinase-1* (PDK1) and AKT (also known as *Protein Kinase B*, *PKB*) on the cell membrane where PDK1 activates AKT by phosphorylation [121, 32].

### AKT-TSC2-Rheb interactions

In a healthy cell, the protein TSC1 stabilises the protein TSC2 by forming the TSC1-TSC2 complex [45, 51]. When AKT gets activated by phosphorylation, it can phosphorylate TSC2, dissociating the TSC1-TSC2 complex. This loss of activity of TSC1-TSC2 is represented by an inhibiting edge from AKT to TSC1-TSC2 in Figure 7.5. Rheb is normally present in an inactive GDP-bound (*guanine diphosphatase*) form in the cells and gets activated in its GTP-bound state. The TSC1-TSC2 complex acts as a GTPase for Rheb (*Ras homologue enriched in brain*), converting Rheb-GTP to Rheb-GDP [88, 105]. This interaction is represented by an inhibiting edge from TSC1-TSC2 to Rheb. Phosphorylated TSC2 loses the GTPase activity towards Rheb, leading to accumulation of GTP-bound Rheb. Activated Rheb then generates pro-growth and anti-apoptotic signals through some additional proteins [1], the details of which we will see in Section 7.1.4.



**Figure 7.6:** Simulation results for AKT module.

### AKT-GSK3-EIF2B interactions

Activated AKT can phosphorylate *glucose synthase kinase 3* (GSK3) leading to inhibition of its kinase activity [122, 71]. This is represented by an inhibiting edge from AKT to GSK3 in Figure 7.5. In normal conditions, un-phosphorylated GSK3 can phosphorylate *initiation factor 2B* (EIF2B) [147]. Phosphorylated EIF2B loses its functionality which promotes mRNA translation initiation in its un-phosphorylated form. This is represented by an inhibiting edge from GSK3 to EIF2B and an activating edge from EIF2B to the node representing cellular growth signals respectively in Figure 7.5.

### AKT-BAD interaction

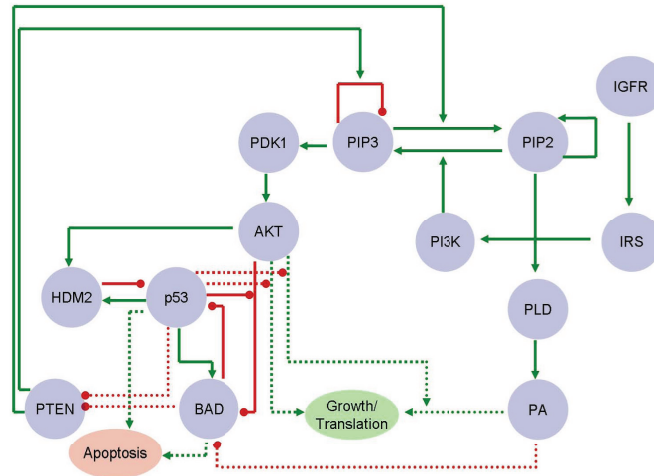
In addition to HDM2, GSK3 and TSC2; AKT can also phosphorylate BAD (which has pro-apoptotic function in its un-phosphorylated form as explained in Section 7.1.1) [146]. This is represented by an inhibiting edge in Figure 7.5.

### p53-sestrin-APMK-TSC2 interactions

Activated p53 transcriptionally activates Sestrin 1 and Sestrin 2 [142, 58] in response to stress. Sestrins recruits AMPK to form a larger complex that can in turn phosphorylate TSC2 [15]. However, unlike AKT phosphorylation of TSC2, AMPK phosphorylation promotes the GTPase activity of TSC2 [15]. This is represented by an activating edge from AMPK to TSC2 in Figure 7.5. Through sestrins and AMPK, a crosstalk is induced between the p53 module and the AKT module.

### AKT module analysis

Since, the “Growth/Translation” node in the GRN of Figure 7.5 has two different input signals, it is modeled at three different activation levels, taking the values: low, medium and high. All the other nodes are modeled at two levels of activation. On simulating the multi-valued GRN in Figure 7.5, we find two attractors. Both attractors show an oscillating “apoptosis” and “growth”.



**Figure 7.7:** PI3K pathway for modeling the cell growth and apoptosis.

Here we display only one attractor in Figure 7.6(a). On simulating the over-expression of AKT, which is often seen in cancer cells, we see high cellular growth and low apoptosis signals (Figure 7.6(b)). This resembles the pro-growth and anti-apoptotic functionality of AKT.

### 7.1.3 PI3K module

The *phosphoinositide 3 kinase* (PI3K) pathway gets activated in response to external growth factors such as insulin and plays a crucial role in regulating the cell growth and apoptosis. Gene mutations in PI3K pathway have been seen in various carcinomas [11, 18, 109] and many drug compounds are currently being investigated to target proteins on this pathway [18]. Figure 7.7 elaborates on the proteins that play a crucial role in the PI3K pathway.

#### IRS activation

The *insulin Receptor Substrate* (IRS) gets activated on phosphorylation by *insulin-like growth factor receptors* (IGFR). IGFR belongs to the family of *receptor protein tyrosine kinase* (RTK). These receptors, sitting across the cell membrane, gets activated on extra-cellular binding of IGFs (*insulin growth factors*). The IGFR undergoes auto-phosphorylation (leading to its activity) and acts as a binding site for IRS. On binding with IGFR, IRS gets phosphorylated and in turn leads to phosphorylation of PI3K. This is represented by activating edges from IRS to PI3K in Figure 7.7.

#### PI3K

The protein PI3K plays an important role in the cell growth mechanism. It gets activated in response to the growth signals (such as glucose and insulin) via



IRS [70, 81] and transfers the growth signal downstream to AKT pathway that plays a major role in cell growth and apoptosis (as discussed in previous section)). PI3K acts as a kinase for *phosphatidylinositol (4,5)-biphosphate* (PIP2) and phosphorylates PIP2 to *phosphatidylinositol (4,5)-triphosphate* (PIP3) [6, 109, 116]. Mutations in PI3K (leading to its constitutive over-expression) has been seen in many cancers [11]. This makes PI3K an interesting target for pharmaceutical research. Inhibitors of PI3K, such as Wortmannin drug and LY294002 have been found to reduce the tumors in various cancer cell lines.

### PIP2-PIP3 balance

Proteins PIP2 and PIP3 are phospholipids and are found on the cell membrane. PIP2, when activated, it can lead to its own production, the details of which we do not consider here and represent it by a self activating edge on PIP2 [96]. PIP2 can be phosphorylated into PIP3 by PI3K [6, 109, 116]. There is a balancing phosphatase called *phosphatase and tensin homolog* (PTEN) that can dephosphorylate PIP3 into PIP2 [59, 109, 116, 134]. PTEN protein is found in almost all tissues in the body and plays a critical role in maintaining a proper balance between PIP2 and PIP3. Over-expression of PI3K or mutation in PTEN (leading to loss of its functionality) can cause abnormally high expression of PIP3 in cells and thereby causing constitutive activation of AKT via PDK1 (Section 7.1.2). This constitutive activation of AKT can lead to abnormal cell growth and attenuates apoptosis.

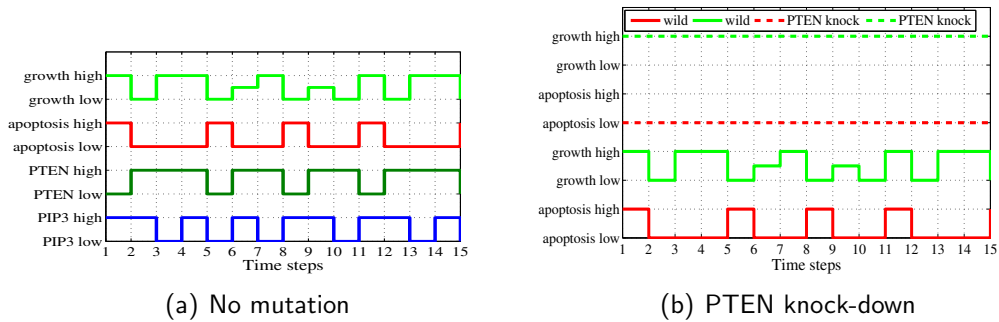
### PIP2-PLD-PA interactions

As will be explained in Section 7.1.4, *phosphatidic acid* (PA) is required for the activation of mTor complex. PA is generated by hydrolysis of phosphatidylcholine to PA and choline. This reaction is catalysed by another protein called *phospholipase D* (PLD) [29]. Activation of PIP2 in turn is required for the activation of PLD [29].

### PI3K module analysis

With the addition of PI3K module to the GRN constructed so far, **genYsis** finds three steady states. All the steady states show the oscillating apoptosis and growth signals. Here we demonstrate only one of the steady states in Figure 7.8(a). As one can see PIP3 and PTEN also displays an oscillating behaviour due to the feedback from p53. In Figure 7.8(b), when PTEN is knocked out the apoptosis signal is completely inhibited and the growth signals stays constitutively high. This behavior represents the pro-growth and anti-apoptotic functionality of PTEN, which is a well known fact from the experiments [72, 25, 109].

The “Growth/Translation” node in Figure 7.7 has three input signals and is modeled at four different activation levels, namely: low, medium, medium-high and high. All the other nodes are modeled at two levels of activation.



**Figure 7.8:** Simulation results for PI3K module.

### 7.1.4 mTOR module

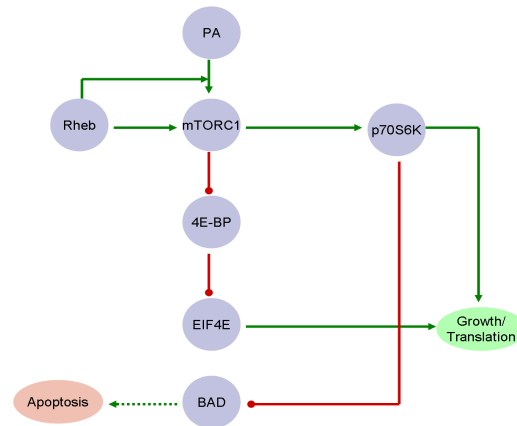
The *mammalian target of rapamycin* (mTor), integrates growth signals, nutrients and energy status and regulates the cell growth, translation and cell apoptosis [30]. Figure 7.9, summarizes the key proteins downstream of mTOR, that helps mTOR in achieving these functionalities.

#### Rheb-PA-mTORC1 interactions

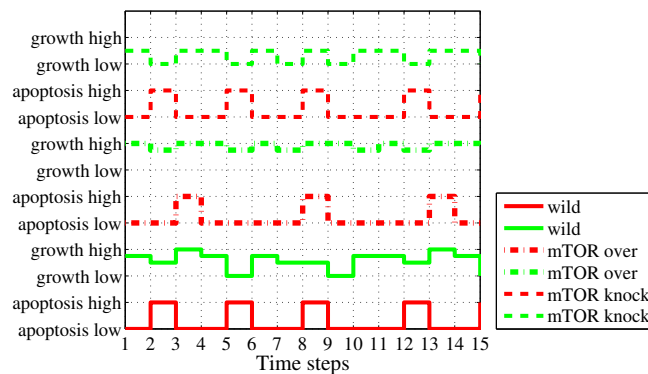
The mTOR complex 1 (mTORC1), or more commonly known as the raptor-mTOR complex, normally lies in an unactivated form. mTORC1 activity is inhibited in the basal state due to the endogenous presence of another protein called FKBP38 inside the cells, which binds and keeps mTORC1 in an inactivated form [152]. mTORC1 can be activated by displacing FKBP38 from its binding site. There are at least two known mechanisms, through Rheb and PA, by which this activation takes place inside the cells. Once Rheb is activated (see Section 7.1.2), it can displace FKBP38 from the mTORC1 complex resulting in the activation of mTORC1 [152]. Under a different mechanism which requires simultaneous presence of both PA and Rheb, Rheb first displaces FKBP38 from its binding site which is then occupied by PA in turn activating the mTORC1 [28, 153, 154]. Rapamycin, a known drug for cancer therapy, is known to occupy the same binding site as FKBP38 and performs anti-cancer action by inactivating the mTORC1 [152, 28, 153, 154]. The similar activation region of rapamycin, Rheb and PA, makes all the three compounds compete for the same site.

#### mTORC1-p70S6K-BAD interactions

The activated mTORC1 in turn activates p70 S6 kinase (S6K) [128], which in turn can phosphorylate the 40S ribosomal protein S6 [52, 128] leading to increased rate of translation of a class of mRNA transcripts (called 5' terminal oligopyrimide mRNA) [61, 42, 53]. This region of mRNA encodes critical components which promotes translation and hence the protein synthesis (which in turn leads to cellular growth). Further, p70S6K is known to inhibit BAD by



**Figure 7.9:** mTor pathway for modeling the cell growth and apoptosis.



**Figure 7.10:** Simulation results for mTOR module.

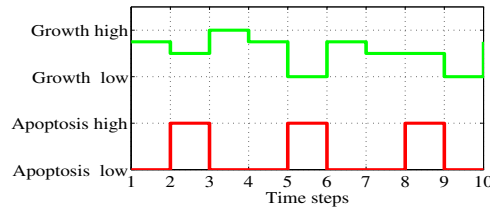
phosphorylation and hence has an inhibiting impact on the cellular apoptosis [52].

### mTORC1-4EBP-EIF4E interactions

EIF4E is a mRNA cap binding protein that regulates mRNA translation [57, 10, 27]. For translation to start, EIF4E must bind with another protein called EIF4G. In normal circumstances, the binding site of EIF4G on EIF4E is occupied by 4E-BP which prevents the translation machinery from getting activated. mTORC1 can remove 4E-BP from its binding site by phosphorylation of 4E-BP. Once 4E-BP is dissociated, EIF4E can form a complex with EIF4G, thereby activating the protein synthesis machinery.

### mTOR module analysis

With the addition of mTOR module to the GRN constructed so far, `genYsis` finds three steady states. All the steady states show the oscillating apoptosis

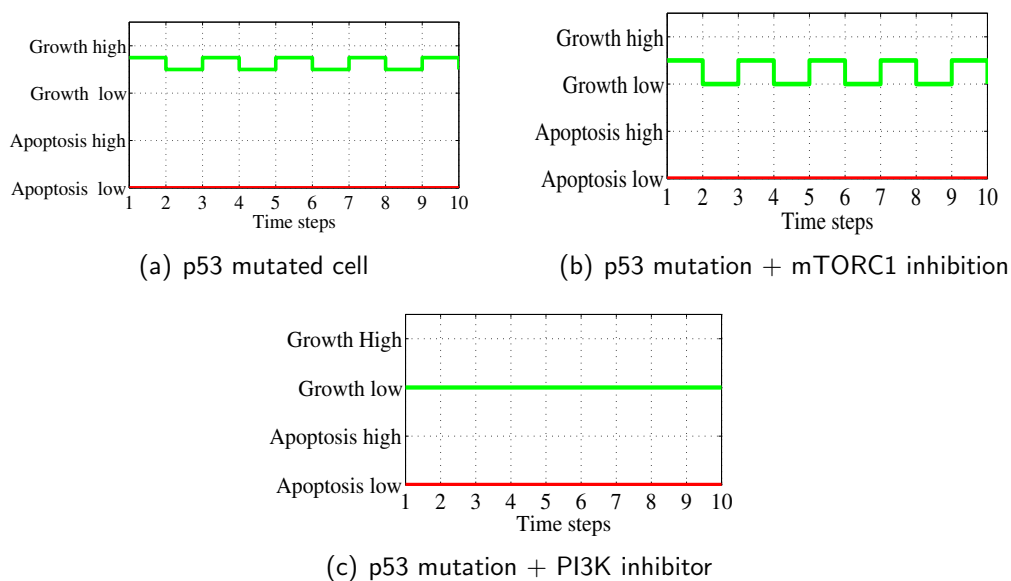


**Figure 7.11:** Simulation results showing steady state expression of growth and apoptosis nodes in the GRN of Figure 7.1. With no mutations, the growth and apoptosis signals show an oscillating behavior indicating the balance between their two extreme values.

and growth signals. Here we demonstrate only one of the steady states in Figure 7.10. The bottom two curves, show the apoptosis and the growth signals in a wild type situation when no gene is mutated. The impact of mTORC1 is more evident in the growth signals (middle two curves in Figure 7.10) which stay constitutively high in the presence of mTORC1 over-expression (which is commonly the case in the cancer cells). When mTORC1 is knocked-down (or inhibited by drugs), growth signal show reduced activity (the top two curves in Figure 7.10) as compared to its wild type activity. All the nodes, except the “Growth/Translation” node are modeled at two activation levels. The “Growth/Translation” has four activation levels, namely: low, medium, medium-high and high.

## 7.2 Modeling growth vs. apoptosis

When the GRN in Figure 7.1 is simulated using *genYsis*, it demonstrates the steady state activity of apoptosis and cell growth signals as shown in Figure 7.11. Apoptosis and cell growth signals show an oscillating behavior demonstrating the ability of the pathway to self-regulate the cellular growth. This self-regulated behavior is possible due to the presence of various negative feedback loops in the GRN, which avoids cell growth and apoptosis signals from constitutive high or constitutive low activity. This balance between the apoptosis and growth signals is known to be skewed in the presence of various gene mutations that lead to abnormal protein activities. Here, we are going to simulate the impact of mutations in p53, PTEN and TSC2 on the growth and apoptosis signals. We will also demonstrate how the imbalance due to these mutations can be restored by perturbing other nodes in the GRN, capturing the mode of action of two commonly used cancer drugs Wortmannin and Rapamycin in cancer therapies. Wortmannin acts as an inhibitor of PI3K protein activity and Rapamycin acts as an inhibitor of mTORC1 protein activity.

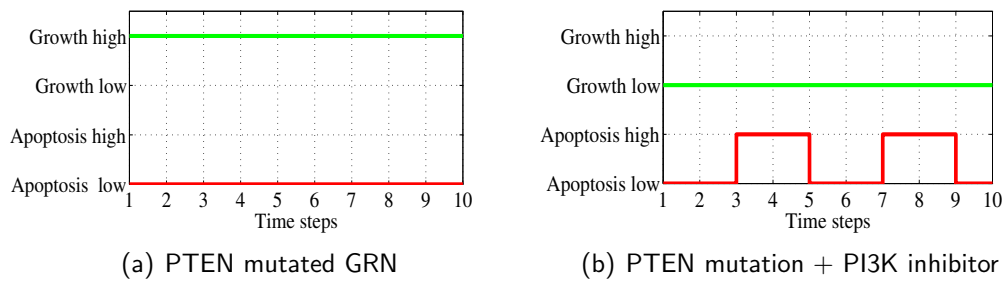


**Figure 7.12:** Simulation results with p53 mutation. (a) p53 is a pro-apoptotic gene and cells can not generate apoptosis signals anymore when p53 loses its functionality due to mutation. (b) p53 mutated cells on inhibiting mTORC1 (by adding inhibitors such as Rapamycin). Cell growth signal in treated cells decreases as compared to untreated cells, but it is not completely inhibited. (c) On treating with PI3K inhibitors (on treatment with drugs such as Wortmannin), the growth signals are completely inhibited.

### 7.2.1 p53 mutation

The tumor suppressor gene p53 plays an important role in various cellular processes such as: (a) halting the cell cycle to avoid cell from proliferating in response to malfunctioning detection, (b) inhibiting the cell growth in response to stress factors so as to avoid burden on scarce resources and (c) repairing the cell machinery if a malfunctioning is detected and guiding the cell to apoptosis if the malfunctioning can not be repaired [31, 13]. Mutant p53, leading to the constitutive inhibition of its activity, is one of the most frequently mutated genes in human cancers [92, 82]. The protein p53 gets activated in response to various stress related factors such as lack of nutrients, hypoxia (lack of oxygen) and UV radiation exposure [31, 13]. Due to its significance in regulating the cell growth, p53 was considered to be a potent target for cancer therapies for a long time. However, so far, restoring the proper level of p53 activity has remained a challenge. The issues in manipulating p53 has pushed the focus of cancer research to other proteins that can regulate the cell growth and apoptosis (and counteract the p53 mutations in the cells) [109].

Figure 7.12(a) shows the simulation results when p53 loses its activity in the GRN. Figures 7.12(b) and 7.12(c) demonstrate the impact of inhibition of mTORC1 and PI3K (in the presence of drug compounds such as Rapamycin



**Figure 7.13:** Simulation results with PTEN mutations. (a) The apoptosis signal is completely inhibited and growth signal stays at the saturating high level. (b) On treatment with PI3K inhibitor, the growth signals can be inhibited.

and Wortmannin respectively) on p53 mutated cells. Rapamycin which can decrease the cell growth (and hence the growth of tumor) is known to be not very effective in low doses (when it selectively targets mTORC1) and has to be either used in high concentrations (resulting in inhibition of various other proteins) or in cocktail with other drug compounds. On the other hand, when the PI3K is inhibited in p53 mutated GRN, the cellular growth signal is completely inhibited (Figure 7.12(c)).

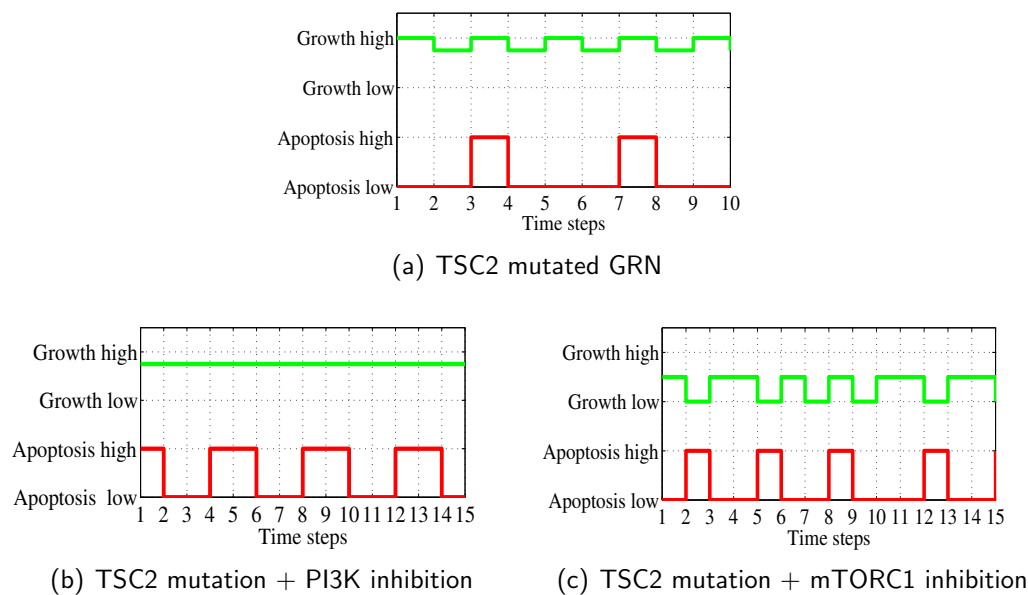
### 7.2.2 PTEN mutation

The protein PTEN is found in almost all tissues in the body and plays a critical role in regulating pro-growth signals. The gene encoding PTEN protein has been found to be frequently mutated (leading to its constitutively high activity) in multiple sporadic tumor types and in patients with cancer predisposition syndromes such as Cowden disease [72, 25, 109].

Figure 7.13(a) demonstrates the growth and apoptosis signals in the presence of mutation in PTEN. PTEN mutation causes both uncontrolled cellular growth and attenuates the apoptosis signal. However, it is known in the literature that the anti-apoptotic and pro-growth behavior of PTEN mutations can be counter-balanced by inhibiting PI3K (using drug compounds such as Wortmannin) [78]. Simulation results in Figure 7.13(b) capture this behavior of PI3K inhibition on PTEN mutated GRN.

### 7.2.3 TSC2 mutation

The *tuberous sclerosis complex* (TSC) is a genetic disorder which can lead to formation of tumors in many different organs, primarily in the brain, eyes, heart, kidney, skin and lungs [126]. Two tumor suppressor genes, TSC1 and TSC2, have been identified for pathogenesis of TSC [21, 150]. In a healthy cell, TSC1 and TSC2 form a complex that acts as an inhibitor of protein translation machinery. In the presence of mutation in either TSC1 or TSC2,



**Figure 7.14:** Simulation results with mutated TSC2. (a) The apoptosis signal is not affected, but the growth signal shows a higher activity as compared to non-mutated GRN. (b) On treatment with PI3K inhibitor, the growth signal shows a decreased activity indicating alternate pathways from PI3K to growth signals. (c) On treatment with mTORC1 inhibitor (such as Rapamycin drug), the constitutively high growth signals can be regulated.

the two proteins can not form a complex, and hence leading to uncontrolled cell growth.

Figure 7.14 demonstrates the simulation results in the presence of TSC2 mutations. As one can see by comparing simulation results in Figure 7.14(a) with Figure 7.13(a), the growth signal does not remain constitutively high and indicates the presence of alternative pathways that can control the cellular growth even in the presence of these mutations. This can also be validated from Figure 7.14(b), which shows a decreased growth signal activity when PI3K is inhibited. Figure 7.14(c) demonstrates the decrease in growth signal on treatment with Rapamycin, indicating the effectiveness of this drug in treatment of TSC2 mutated carcinomas.

The difference in growth signal activities in PTEN mutations (Figure 7.13(a)) vs. TSC2 mutations (Figure 7.14(a)) also conform with the observation made in the literature that Cowden disease, which is associated with PTEN mutations, is characterized by a much higher cancer risk than is tuberous sclerosis complex disease (associated with TSC2 mutations) [137].

*In silico* simulations, such as the ones shown in this section, can be efficiently performed using algorithms proposed in this thesis and can be very useful for biologists to study the crosstalk among various genes (and proteins) in a complex pathway. Even though the simulation results in this section show

effectiveness of one drug over another drug, it is often important to study the concomitant effect of two or more drug compounds. For instance, PI3K activity is important for insulin signaling, metabolism and normal brain function [40]. So inhibition of PI3K can lead to various other abnormalities such as diabetes, schizophrenia and bipolar disorder [136, 40, 149]. This necessitates modeling the multiple inhibitions in a GRN that can capture such diverse effects of drug therapies. The GRN in Figure 7.1 is a small subset of hundreds of genes and proteins that are known to play a crucial role in a day to day functioning of a cell and needs to be explored further. Nonetheless, it captures some important experimental results known in the literature.

### 7.3 Summary

In this chapter, we provided an application of algorithms proposed in previous chapters for modeling GRNs. A GRN was constructed for modeling the signaling pathways that maintain a balance between the pro-growth and anti-growth signals in a healthy cell. By simulating various known cancer mutations and their corresponding drug targets, the GRN has been shown to represent the known experimental outcomes well. Furthermore, an evolutionary approach for constructing a GRN has been demonstrated that uses **genYsis** to ensure to ensure that local biological functionalities are conserved (i.e. global behaviour of the GRN is preserved) as new proteins are added to a GRN. Even though the interactions in the GRN, shown in this chapter, were manually inferred from literature, this process can potentially be automated using the literature mining and machine learning algorithms. **GenYsis** can prove invaluable in analysing the inferred GRN and to ensure that the inferred network correctly represents the dynamics of the biological phenomena under investigation.



---

# Conclusions

---

# 8

In this thesis, I have proposed algorithms to model various aspects of GRNs. The formalism proposed in different chapters can be seen as different stages of construction and modeling of GRNs. Boolean modeling in Chapter 3 requires a very small amount of prior knowledge about the biological system being modeled. As we advance through multiple valued rules in Chapter 4, more refined information about gene and protein regulation is accommodated in the modeling framework. The PBNs in Chapter 5 provide a mechanism to study the GRNs resulting from the previous chapters in the context of the experimental data. Finally, when one is confident that the constructed GRN resembles the dynamics of the biological system under investigation, robustness and stochastic simulations can be performed on the resulting GRN.

## 8.1 Thesis summary and contributions

Chapters 1 and 2 were dedicated to the introduction of the problem and the background knowledge on GRNs respectively. Chapters 3 to 7 gave the original contributions of our research. Contributions of this thesis are summarised as follows:

- **Chapter 3** formally introduced the problem of modeling the dynamics in GRNs using Boolean algebra and implicit representation of finite state machines. The formalism proposed in this chapter enables computation of steady states and dynamical attractors of large GRNs with over thousand nodes. Further, under this formalism, it is possible to model different transition schemes such as synchronous and asynchronous, by slightly modifying the underlying Boolean functions but using the same algorithms. Although the asynchronous dynamic models are closer to bi-

ological phenomena than the synchronous models, their computational overhead can be prohibitively large for some GRNs. It was demonstrated that the computation time for identifying all attractors in the asynchronous model can be reduced by exploiting similarities between attractors of the synchronous and the asynchronous models. Based on these algorithms for computing steady states, further algorithms were developed for modeling multiple gene perturbation experiments. These algorithms can be used for *in silico* simulation of experimental protocols which involve application of different stimuli on a cell culture over a range of time. The applicability of algorithms was shown by modeling the T-Helper cell GRN. It was demonstrated how T-helper cells can differentiate into Th0, Th1 and Th2 cell types when subjected to gene perturbations.

- **Chapter 4** extended the formalism proposed in Chapter 3 to model multi-valued GRNs. In the multi-valued modeling of GRNs a gene (or protein) can exist at multiple levels of activation, such as: low, medium and high. Multiple levels of activation of a gene (or protein) can be sometimes essential as the same gene (or protein) may show different functionalities at different levels of activity. This extension to formalism proposed in Chapter 3 was executed by treating each activation level of the gene as a different Boolean variable. Additional Boolean constraints were used to define a relationship between the set of Boolean variables that correspond to the activity levels of the same gene (or protein). Under the modified formalism, algorithms presented in Chapter 3 could be employed to compute the steady states of multi-valued GRNs. Further, a sigmoid function was introduced to extract multiple level transition rules from the Boolean transition functions (i.e. activating and inhibiting edges) of GRNs. This automatic extracting of rules has the benefit that one does not have to explicitly identify transition rules for the each activation level of a gene from the literature or from the experimental data. Finally, a combined Boolean-ODE methodology was introduced based on sigmoid functions where the Boolean attractors are fed into a numerical solver of ODEs to compute the continuous counterpart of Boolean steady states.
- **Chapter 5** presented an extension of Boolean modeling of GRNs to non-deterministic networks where multiple alternative biological functions can cause the activation of a gene (or protein). Alternate functions have respective probabilities of selection leading to a stochastic FSM. PBNs have been proposed in the past for modeling the dynamics in such a non-deterministic representation of GRNs using markov chains based explicit approaches. Under such approaches, the steady state probability distribution is assumed to correspond to the cellular state. However, a GRN can have multiple steady states and hence, depending upon the

starting probability distribution, there can be multiple stationary distributions in a given markov chain representation. Further, size of the probability transition matrix required for explicit computation of stationary distribution can be astronomically large due to the exponential large state space of a GRN. I addressed these issues in this chapter by extending the implicit representation of deterministic Boolean networks to PBNs. Unlike in the explicit methods, the problem of computation of stationary distribution was divided into two problems. First, all attractors of the stochastic FSM were identified using the algorithms proposed in Chapter 3 and then in the second problem, the steady state probability distribution of each attractor is computed. By dividing the problem into two sub-problems, I address the issue of non-unique stationary distribution of the markov chains. Further, the complexity arising due to exponential state space is addressed by using implicit representation techniques based on ROBDDs and ADDs. The computational efficiency and functionalities of algorithms proposed in this chapter was shown by applying them on various synthetic GRNs and on a GRN proposed in the literature for modeling a specific cancer type called gliomas.

- **Chapter 6** extends Boolean formalism proposed in Chapter 3 to incorporate inherent stochasticity in biological phenomena. A gene or protein can be normally activated by more than one biological phenomena. This redundancy is inherent in biology to avoid a complete breakdown of the system in the event of malfunctioning in a single gene (or protein) regulation phenomena. However, every gene regulation phenomena has a probability of failure which is normally reflected by the complexity of the regulation mechanism itself. In our Boolean modeling approach, the gene (or protein) regulation mechanisms are represented by Boolean functions. To model stochasticity in gene regulation mechanisms, a fault modeling approach called *stochasticity in functions* (SIF) was introduced in this chapter. In SIF, Boolean functions have a probability of failure reflecting the stochasticity (and hence complexity) of underlying biological functions. The SIF model was shown to give more biological realistic results as compared to existing approaches for modeling stochasticity in GRNs. By applying SIF on T-Helper and T-Cell receptor GRN, it was demonstrated that while the two GRNs will be declared non-robust to stochasticity for a small amount of noise under the existing models, SIF can efficiently model the impact of gradually increasing noise on the robustness of GRNs.
- **Chapter 7** introduced a case study on the construction and modeling of a GRN that represents a balance between cellular growth and apoptosis signals in a healthy vs. a cancer cell. The GRN was constructed incrementally, starting from a small set of four proteins. In each iteration of network construction, the GRN was expanded by adding new regulators

and targets of base proteins. Algorithms proposed in this thesis were then applied to ensure that the expanded GRN correctly reflects the intended end result (i.e a balance between apoptosis and cell growth). The GRN resulting from the final iteration was then modeled for various known gene mutations that are known to cause an imbalance between cell growth and apoptosis. The impact of gene mutations on unregulated cell growth was demonstrated reflecting the critical role of these genes (or proteins) in the formation of tumors. Further, the therapeutic targets to restore the imbalance in mutated cells were modeled reflecting the impact of commonly used drugs. The bottom-up GRN construction mechanism shown in this chapter displays one possible application of our modeling toolbox `genYsis` which can be potentially automated in the future by using it in combination with data and literature mining algorithms.

## 8.2 Future work

The formalism proposed in this thesis can be extended in various interesting directions.

In this thesis, I focussed on modeling the dynamics of single cells. However, cells can also communicate with each other by secreting proteins that other cells can sense at their surface. To model the influence of inter-cell communication on the population of cells, it will be interesting to generalise the algorithms such that multiple copies of GRNs are simulated at the same time. Cell population can be seen as a swarm of cells where each cell has a computation mechanism specified by its own GRN and cells communicate via expression of a few nodes that are part of their GRN. Such a modeling mechanism can link the simulation of the GRN of a single cell with the GRN of its neighbouring cells. This modeling approach will have an advantage over the more traditional ODE based approaches where protein expression is normalised over the population of cells assuming that all cells behave in an identical way and show concurrent dynamics.

Algorithms proposed in this thesis can also benefit by including parameters that reflect the notion of time in the gene regulation mechanisms and parameters to reflect probability of failure of a biological function. However in practice, it can be difficult to learn these parameters from literature or estimate them from the experimental data. Structural biology can provide some of the information that can potentially be translated into these parameters. For example, it has been demonstrated that the length of protein coding region has an impact on the time it takes for a protein to be expressed inside a cell from the time the transcription factor is activated [68]. Eventhough such information is readily available in databases, it has not been applied in the context of the dynamics of a GRN. On the other hand, in order to assign a probability of failure to each biological function, information on binding sites,

binding affinity and number of binding agents required to activate a protein can be effectively put to use. These extensions, however, will require a detailed study to translate the above biological characteristics into the corresponding equation parameters.

As the experimental tools available to biologists will become more advanced, additional insights into gene and protein regulation mechanisms will become available. However, with this improved understanding of biological phenomena, there will be a desire to computationally model increasingly complex biological systems. To deal with the complex biological systems, various abstraction mechanisms will have to be employed in the computational tools. Therefore, even though the formalism proposed in this thesis seems to be an abstraction of actual biological mechanisms, computational methods such as the ones proposed in this thesis will form the basis of modeling in biology in the future. I believe that the discrete modeling and analysis techniques proposed in this thesis will find more applications as the complexity of the systems that one would like to computationally model grows and discrete modeling will be the direction of research in systems biology in the coming years.



# Bibliography

---

- [1] Astrinidis A and Henske EP. “Tuberous sclerosis complex: linking growth and energy signaling pathways with human disease”. *Oncogene*, 24:7475–7481, 2005.
- [2] Becskei A and Serrano L. “Engineering stability in gene networks by autoregulation”. *Nature*, 405:590–593, 2000.
- [3] Fauré A, Naldi A, Chaouiya C, and Thieffry D. “Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle”. *Bioinformatics*, 22:e124–131, 2006.
- [4] Levchenko A. “Dynamical and integrative cell signaling: challenges for the new biology”. *Biotechnology Bioengineering*, 84:773–782, 2003.
- [5] Semenov A and Yakovlev A. “Verification of asynchronous circuits using time Petri net unfolding”. *Proceedings of the 33rd annual conference on Design automation, Las Vegas 1996*, pages 59–62, 1996.
- [6] Toker A and Cantley LC. “Signalling through the lipid products of phosphoinositide-3-OH kinase”. *Nature*, 387:673–676, 1997.
- [7] Xie A and Beerel PA. “Efficient state classification of finite state markov chains”. *Proceedings of Design Automation Conference*, 1998.
- [8] Yakovlev A, Semenov A, Koelmans AM, and Kinniment DJ. “Petri nets and asynchronous circuit design”. *IEEE colloquium on Design and Test Asynchronous Systems*, pages 8/1–8/6, 1996.
- [9] Yates A, Bergmann C, Van Hemmen JL, Stark J, and Callard R. “Cytokine-modulated regulation of helper T cell populations”. *Journal of Theoretical Biology*, 206:539–560, 2000.
- [10] Gingras AC, Gygi SP, Raught B, Polakiewicz RD, Abraham RT, Hoekstra MF, Aebersold R, and Sonenberg N. “Regulation of 4E-BP1 phosphorylation: a novel two-step mechanism”. *Genes and Development*, 13:1422–1437, 1999.

- [11] Bader AG, Kang S, Zhao L, and Vogt PK. “Oncogenic PI3K deregulates transcription and translation”. *Nature Reviews Cancer*, 5:921–929, 2005.
- [12] Dr aghici S. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC, Florida, 2003.
- [13] Levine AJ. “p53, the cellular gatekeeper for growth and division”. *Cell*, 88:323–331, 1997.
- [14] Ribeiro AS and Kauffman SA. “Noisy attractors and ergodic sets in models of gene regulatory networks”. *Journal of Theoretical Biology*, 247:743–755, 2007.
- [15] Budanov AV and Karin M. “p53 target genes sestrin1 and sestrin2 connect genotoxic stress and mTOR signaling”. *Cell*, 134:451–460, 2008.
- [16] Alberts B, Johnson A, Lewis J, Raff M, Roberts K, and Walter P. *Molecular Biology of the Cell*. Garland Science, 2007.
- [17] Goodwin BC. *Temporal organization in cells; A dynamic theory of cellular control processes*. Academic Press, New York., 1963.
- [18] Hennessy BT, Smith DL, Ram PT, Lu Y, and Mills GB. “Exploiting the PI3K/AKT pathway for cancer drug discovery”. *Nature Reviews Drug Discovery*, 4:988–1004, 2005.
- [19] Bergmann C, van Hemmen JL, and Segel LA. “Th1 or Th2: how an appropriate T helper response can be made”. *Bulletin of Mathematical Biology*, 63:405–430, 2001.
- [20] Espinosa-Soto C, Padilla-Longoria P, and Alvarez-Buylla ER. “A gene regulatory network model for cell fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles”. *Plant Cell*, 16:2923–2939, 2004.
- [21] Consortium. “Identification and characterization of the tuberous sclerosis gene on chromosome 16”. *Cell*, 75:1305–1315, 1993.
- [22] Rao CV, Wolf DM, and Arkin AP. “Control, exploitation and tolerance of intracellular noise”. *Nature*, 421:231–237, 2002.
- [23] Agnello D, Lankford CS, Bream J, Morinobu A, Gadina M, O’Shea JJ, and Frucht DM. “Cytokines and transcription factors that regulate T helper cell differentiation: new players and new insights”. *Journal of Clinical Immunology*, 23:147–162, 2003.
- [24] Gonze D and Goldbeter A. “Circadian rhythms and molecular noise”. *Chaos*, 16:026110, 2006.



- [25] Liaw D, Marsh DJ, Li J, Dahia PL, Wang SI, Zheng Z, Bose S, Call KM, Tsou HC, Peacocke M, Eng C, and Parsons R. “Germline mutations of the PTEN gene in Cowden disease, an inherited breast and thyroid cancer syndrome”. *Nature Genetics*, 16:64–67, 1997.
- [26] Schultz D, Jacob EB, Onuchic JN, and Wolynes PG. “Molecular level stochastic model for competence cycles in *Bacillus subtilis*”. *Proceedings of National Academy of Science of the USA*, 104:17582–17587, 2007.
- [27] Shahbazian D, Roux PP, Mieulet V, Cohen MS, Raught B, Taunton J, Hershey JWB, Blenis J, Pende M, and Sonenberg N. “The mTOR/PI3K and MAPK pathways converge on eIF4B to control its phosphorylation and activity”. *EMBO Journal*, 25:2781–2791, 2006.
- [28] Foster DA. “Regulation of mTOR by phosphatidic acid?”. *Cancer Research*, 67:1–4, 2007.
- [29] Foster DA and Xu L. “Phospholipase D in cell proliferation and cancer”. *Molecular Cancer Research*, 1:789–800, 2003.
- [30] Sarbassov DD, Ali SM, and Sabatini DM. “Growing roles for the mTOR pathway”. *Current Opinion in Cell Biology*, 17:596–603, 2005.
- [31] Lane DP. “p53, guardian of the genome”. *Nature*, 358:15–16, 1992.
- [32] Alessi DR, Andjelkovic M, Caudwell B, Cron P, Morrice N, Cohen P, and Hemmings BA. “Mechanism of activation of protein kinase B by insulin and IGF-1”. *EMBO Journal*, 15:6541–6551, 1996.
- [33] Gillespie DT. “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions”. *Journal of Computational Physics*, 22:403–434, 1976.
- [34] Gillespie DT. “Exact stochastic simulation of coupled chemical reactions”. *Journal of Physical Chemistry*, 81:2340–2361, 1977.
- [35] Coen E and Meyerowitz EM. “The war of the whorls: genetic interactions controlling flower development”. *Nature*, 353:31–37, 1991.
- [36] Remy E, Ruet P, Mendoza L, Thieffry D, and Chaouiya C. “From logical regulatory graphs to standard Petri nets: Dynamical roles and functionality of feedback circuits”. *Lecture Notes in Computer Science*, 4230: 56–72, 2006.
- [37] Davidson EH, Rast JP, Oliveri P, Ransick A, Caletani C, Yuh CH, Minokawa T, Amore G, Hinman V, Arenas-Mena C, Otim O, Brown CT, Livi CB, Lee PY, Revilla R, Rust AG, Pan Z, Schilstra MJ, Clarke PJ, Arnone MI, Rowen L, Cameron RA, McClay DR, Hood L, and Bolouri H. “A genomic regulatory network for development”. *Science*, 295:1669–78, 2002.

- [38] Snoussi EH and Thomas R. “Logical identification of all steady states : the concept of feedback loop-characteristic states.”. *Bulletin of Mathematical Biology*, 55:973–991, 1993.
- [39] Alvarez-Buylla ER, Chaos A, Aldana M, Bentez M, Cortes-Poza Y, Espinosa-Soto C, Hartasanchez DA, Lotto RB, Malkin D, Escalera Santos GJ, and Padilla-Longoria P. “Floral Morphogenesis: Stochastic explorations of a gene network epigenetic landscape”. *PLoS ONE*, 3:e3626, 2008.
- [40] Emamian ES, Hall D, Birnbaum MJ, Karayiorgou M, and Gogos JA. “Convergent evidence for impaired AKT1-GSK $\beta$  signaling in schizophrenia”. *Nature Genetics*, 36:131–137, 2004.
- [41] Li F, Long T, Lu Y, Ouyang Q, and Tang C. “The yeast cell-cycle network is robustly designed”. *Proceedings of National Academy of Science of the USA*, 101:4781–4786, 2004.
- [42] Loreni F, Thomas G, and Amaldi F. “Transcription inhibitors stimulate translation of 5' TOP mRNAs through activation of S6 kinase and the mTOR/FRAP signalling pathway”. *European Journal of Biochemistry*, 267:6594–6601, 2000.
- [43] Somenzi F. “CUDD: CU Decision Diagram Package Release 2.4.1”. *University of Colorado at Boulder*, 2005.
- [44] Yang J.C-Y Fujita M, McGeer PC. “Multi-Terminal Binary Decision Diagrams: An efficient datastructure for matrix representation”. *Formal Methods in System Design*, 10:149–169, 1997.
- [45] Benvenuto G, Li S, Brown SJ, Braverman R, Vass WC, Cheadle JP, Halley DJ, Sampson JR, Wienecke R, and DeClue JE. “The tuberous sclerosis-1 (TSC1) gene product hamartin suppresses cell growth and augments the expression of the tsc2 product tuberlin by inhibiting its ubiquitination”. *Oncogene*, 19:6306–6316, 2000.
- [46] De Micheli G. *Synthesis and optimization of digital circuits*. Mc Graw-Hill Higher Education, 2009.
- [47] MacBeath G and Schreiber SL. “Printing proteins as microarrays for high-throughput function determination”. *Science*, 289:1760–1763, 2000.
- [48] Weisbuch G, DeBoer RJ, and Perelson AS. “Localized memories in idiotypic networks”. *Journal of Theoretical Biology*, 146:483–499, 1990.
- [49] Fuller GN, Rhee CH, Hess KR, Caskey LS, Wang R, Bruner JM, Yung WKA, and Zhang W. “Reactivation of Insulin-like Growth Factor Binding Protein 2 Expression in Glioblastoma Multiforme”. *Cancer Research*, 59:4228–4232, 1999.

- [50] Krueger GR, Marshall GR, Junker U, Schroeder H, and Buja LM. “Growth factors, cytokines, chemokines and neuropeptides in the modeling of T-cells”. *In Vivo*, 17:105–118, 2002.
- [51] Chong-Kopera H, Inoki K, Li Y, Zhu T, Garcia-Gonzalo FR, Rosa JL, and Guan KL. “TSC1 stabilizes TSC2 by inhibiting the interaction between TSC2 and the HERC1 ubiquitin ligase”. *Journal of Biological Chemistry*, 281:8313–8316, 2006.
- [52] Harada H, Andersen JS, Mann M, Terada N, and Korsmeyer SJ. “p70S6 kinase signals cell survival as well as growth, inactivating the proapoptotic molecule BAD”. *Proceedings of National Academy of Science of the USA*, 98:9666–9670, 2001.
- [53] Heinonen H, Nieminen A, Saarela M, Kallioniemi A, Klefstrm J, Hautaniemi S, and Monni O. “Deciphering downstream gene targets of PI3K/mTOR/p70S6K pathway in breast cancer”. *BMC Genomics*, 9, 2008.
- [54] Kitano H. “Computational systems biology”. *Nature*, 420:206–210, 2002.
- [55] Kitano H. “Systems biology: a brief overview”. *Science*, 295:1662–1664, 2002.
- [56] Maamar H, Raj A, and Dubnau D. “Noise in gene expression determines cell fate in *Bacillus subtilis*”. *Science*, 317:526–529, 2007.
- [57] Matsuo H, Li H, McGuire AM, Fletcher CM, Gingras AC, Sonenberg N, and Wagner G. “Structure of translation factor EIF4E bound to m7GDP and interaction with 4E-binding protein”. *Natural Structural and Molecular Biology*, 4:717–724, 1997.
- [58] Peeters H, Debeer P, Bairoch A, Wilquet V, Huysmans C, Parthoens E, Fryns JP, Gewillig M, Nakamura Y, Niikawa N, Van de Ven W, and Devriendt K. “PA26 is a candidate gene for heterotaxia in humans: identification of a novel PA26-related gene family in human and mouse”. *Human Genetics*, 112:573–580, 2003.
- [59] Sun H, Lesche R, Li DM, Liliental J, Zhang H, Gao J, Gavrilova N, Mueller B, Liu X, and Wu H. “PTEN modulates cell cycle progression and cell survival by regulating phosphatidylinositol 3,4,5,-trisphosphate and Akt/protein kinase B signaling pathway”. *Proceedings of National Academy of Science of the USA*, 96:6199–6204, 1999.
- [60] Zou H, Yang R, Hao J, Wang J, Sun C, and Fesik SW. “Regulation of the Apaf-1/Caspase-9 apoptosome by Caspase-3 and XIAP”. *Journal of Biological Chemistry*, 278:8091–8098, 2003.

- [61] Jefferies HBJ, Fumagalli S, Dennis PB, Reinhard C, Pearson RB, and Thomas G. “Rapamycin suppresses 5’TOP mRNA translation through inhibition of p70s6k”. *EMBO Journal*, 16:3693–3704, 1997.
- [62] McAdams HH and Arkin A. “Its a noisy business! Genetic regulation at the nanomolar scale”. *Trends in Genetics*, 15:65–69, 1999.
- [63] Touati HJ, Savoj H, Lin B, Brayton RK, and Sangiovanni-Vincentelli A. “Implicit state enumeration of finite-state machines using BDDs”. *Proceedings of ICCAD’90*, 1990.
- [64] Andersen HR. “An introduction to Binary Decision Diagrams”. *Lecture notes for 49285 Advanced Algorithms E97*, 1997.
- [65] Shmulevich I, Dougherty ER, Kim S, and Zhang W. “Probabilistic Boolean Networks: a rule-based uncertainty model for gene regulatory network”. *Bioinformatics*, 18:261–274, 2002.
- [66] Shmulevich I and Lahdesmaki H. “PBN Matlab Toolbox”. URL [personal.systemsbio.net/ilya/](http://personal.systemsbio.net/ilya/).
- [67] Shmulevich I, Gluhovsky I, Hashimoto RF, Dougherty ER, and Zhang W. “Steady-State Analysis of Genetic Regulatory Networks Modelled by Probabilistic Boolean Networks”. *Comparative and Functional Genomics*, 4:601–608, 2003.
- [68] Swinburne IA, Miguez DG, Landgraf D, and Silver PA. “Intron length increases oscillatory periods of gene expression in animal cells”. *Genes and Development*, 22:2342–2346, 2008.
- [69] DeRisi J, Penland L, Brown PO, Bittner ML, Meltzer PS, Ray M, Chen Y, Su YA, and Trent JM. “Use of a cDNA microarray to analyse gene expression patterns in human cancer”. *Nature Genetics*, 14:457–460, 1996.
- [70] Lackey J, Barnett J, Davidson L, Batty IH, Leslie NR, and Downes CP. “Loss of PTEN selectively desensitizes upstream IGF1 and insulin signaling”. *Oncogene*, 26:7132–7140, 2007.
- [71] Lee J and Kim MS. “The role of GSK3 in glucose homeostasis and the development of insulin resistance”. *Diabetes research and Clinical Practice*, 77:S49–S57, 2007.
- [72] Li J, Yen C, Liaw D, Podsypanina K, Bose S, Wang SI, Puc J, Miliareisis C, Rodgers L, McCombie R, Bigner SH, Giovanella BC, Ittmann M, Tycko B, Hibshoosh H, Wigler MH, and Parsons R. “PTEN, a putative protein tyrosine phosphatase gene mutated in human brain, breast, and prostate cancer.”. *Science*, 275:1943–1947, 1997.

- [73] Piette J, Neel H, and Marchal V. “Mdm2: keeping p53 under control”. *Oncogene*, 15:1001–1010, 1997.
- [74] Saez-Rodriguez J, Simeoni L, Lindquist JA, Hemenway R, Bommhardt U, Arndt B, Haus UU, Weismantel R, Gilles E D, Klamt S, and Schraven B. “A logical model provides insights into T Cell Receptor signaling”. *PLoS Computational Biology*, 3:e163, 2007.
- [75] Smith J, Theodoris C, and Davidson EH. “A gene regulatory network subcircuit drives a dynamic pattern of gene expression”. *Science*, 318: 794–797, 2007.
- [76] Torres J, Rodriguez J, Myers MP, Valiente M, Graves JD, Tonks NK, and Pulido R. “Phosphorylation-regulated cleavage of the tumor suppressor PTEN by Caspase-3.”. *Journal of Biological Chemistry*, 278: 30652–30660, 2003.
- [77] Yu J, Zhang L, Hwang PM, Kinzler KW, and Vogelstein B. “PUMA induces the rapid apoptosis of colorectal cancer cells.”. *Molecular Cell*, 7:673–682, 2001.
- [78] Zhang J, Choi Y, Mavromatis B, Lichtenstein A, and Li W. “Preferential killing of PTEN-null myelomas by PI3K inhibitors through Akt pathway.”. *Oncogene*, 22:6289–6295, 2003.
- [79] Le Boudec J-Y. “Modelling the Immune System Toolbox: Stochastic Reaction Models”. *Course material: Modeling the Immune System*, 2007.
- [80] Burch JB, Clarke EM, and Long DE. “Symbolic model checking with partitioned transition relations”. *Proceedings of International Conference on VLSI’91*, 1991.
- [81] Easton JB, Kurmasheva RT, and Houghton PJ. “IRS-1: auditing the effectiveness of mtor inhibitors”. *Cancer Cell*, 9:153–155, 2006.
- [82] Bourdon JC. “p53 and its isoforms in cancer”. *British Journal of Cancer*, 97:277–282, 2007.
- [83] Venter JC, Adams MD, and Myers EW et al. “The sequence of the human genome”. *Science*, 291:1304–1351, 2001.
- [84] Chipuk JE, Kuwana T, Bouchier-Hayes L, Droin NM, Newmeyer DD, Schuler M, and Green DR. “Direct activation of Bax by p53 mediates mitochondrial membrane permeabilization and apoptosis”. *Science*, 303: 1010–1014, 2004.
- [85] Joung JK, Ramm EI, and Pabo CO. “A bacterial two-hybrid selection system for studying protein-DNA and protein-protein interactions”. *Proceedings of National Academy of Science of the USA*, 97:7382–7387, 2000.

- [86] Bowman JL, Smyth DR, and Meyerowitz EM. “Genes directing flower development in *arabidopsis*.”. *Plant Cell*, 1:37–52, 1989.
- [87] Pedraza JM and Oudenaarden AV. “Noise propagation in gene networks”. *Science*, 307:1965–1969, 2005.
- [88] Inoki K, Li Y, Xu T, and Guan KL. “Rheb GTPase is a direct target of TSC2 GAP activity and regulates mTOR signaling”. *Genes and Development*, 17:1829–1834, 2003.
- [89] Nakano K and Vousden KH. “PUMA, a novel proapoptotic gene, is induced by p53”. *Molecular Cell*, 7:683–694, 2001.
- [90] Willadsena K and Wiles J. “Robustness and state-space structure of Boolean gene regulatory models”. *Journal of Theoretical Biology*, 249: 749–765, 2007.
- [91] Chen KC. “Integrative analysis of cell cycle control in budding yeast”. *Molecular Biology of Cell*, 15:3841–3862, 2004.
- [92] Vousden KH and Lane DP. “p53 in health and disease”. *Nature Reviews Molecular Cell Biology*, 8:275–283, 2007.
- [93] Young KH. “Yeast two-hybrid: so many interactions, (in) so little time....”. *Biology of reproduction*, 58:302–311, 1998.
- [94] Murphy KM and Reiner SL. “The lineage decisions on helper T cells”. *Nature Review Immunology*, 2:933–944, 2002.
- [95] Benjamin L, Cassimeris L, Lingappa VR, and Plopper G. *Cells*. Jones and Bartlett Publishers, 2007.
- [96] Gervais L, Claret S, Januschke J, Roth S, and Guichet A. “PIP5K-dependent production of PIP2 sustains microtubule organization to establish polarized transport in the drosophila oocyte”. *Development*, 135: 3829–3838, 2008.
- [97] Hood L. “Systems biology: integrating technology, biology, and computation”. *Mechanisms of Ageing and Development*, 124:9–16, 2002.
- [98] Kadanoff L, Coppersmith S, and Aldana M. “Boolean dynamics with random couplings”. *Springer Applied Mathematical Sciences Series*, Special volume:23–89, 2003.
- [99] Mendoza L. “A network model for the control of the differentiation process in Th cells”. *Biosystems*, 84:101–114, 2005.
- [100] Mendoza L, Thieffry D, and Alvarez-Buylla ER. “Genetic control of flower morphogenesis in *Arabidopsis thaliana*: a logical analysis”. *Bioinformatics*, 15:593–606, 1999.

- [101] Mendoza L and Alvarez-Buylla ER. “Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis”. *Journal of Theoretical Biology*, 193:307–319, 1998.
- [102] Mendoza L and Xenarios I. “A method for the generation of standardized qualitative dynamical systems of regulatory networks”. *Theoretical Biology and Medical Modeling*, 3, 2006.
- [103] Mayo LD and Donner DB. “A phosphatidylinositol 3-kinase/Akt pathway promotes translocation of Mdm2 from the cytoplasm to the nucleus”. *Proceedings of National Academy of Science of the USA*, 98: 11598–11603, 2001.
- [104] Mayo LD and Donner DB. “p53-Mdm2-the affair that never ends”. *Carcinogenesis*, 23:541–547, 2002.
- [105] Saucedo LJ, Gao X, Chiarelli DA, Li L, Pan D, and Edgar BA. “Rheb promotes cell growth as a component of the insulin/TOR signalling network”. *Nature Cell Biology*, 5:566–571, 2003.
- [106] Steggles LJ, Banks R, Shaw O, and Wipat A. “Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach.”. *Bioinformatics*, 23(3):336–343, 2007.
- [107] D Lockhart, H Dong, M Byrne, M Follettie, M Gallo, M Chee, M Mittmann, C Wang, M Kobayashi, H Horton, , and E L Brown. “Expression monitoring by hybridization to high-density oligonucleotide arrays”. *Nature Biotechnology*, 14:1675–1680, 1996.
- [108] Baldamus M and Schneider K. “The BDD space complexity of different forms of concurrency”. *Proceedings of ICACSD '01*, 2001.
- [109] Cully M, You H, Levine AJ, and Mak TW. “Beyond PTEN mutations: the PI3K pathway as an integrator of multiple inputs during tumorigenesis”. *Nature Reviews Cancer*, 6:184–192, 2006.
- [110] Heiner M and Koch I. “Petri net based model validation in systems biology”. *J. Cortadella and W. Reisig (Eds), ICATPN04, LNCS*, 3099: 216–237, 2004.
- [111] Kaern M, Elston TC, Blake WJ, and Collins JJ. “Stochasticity in gene expression: From theories to phenotypes”. *Nature Reviews Genetics*, 6: 451–464, 2005.
- [112] Vogt M, Butz K, Dymalla S, Semzow J, and Hoppe-Seyler F. “Inhibition of Bax activity is crucial for the antiapoptotic function of the human papillomavirus E6 oncoprotein”. *Oncogene*, 25:4009–4015, 2006.

- [113] Elowitz MB, Levine AJ, Siggia ED, and Swain PS. “Stochastic gene expression in a single cell”. *Science*, 297:1183–1186, 2002.
- [114] Davidich MI and Bornholdt S. “Boolean network model predicts cell cycle sequence of fission yeast”. *PLoS ONE*, 3:e1672, 2008.
- [115] DeyHoles MK and Sieburth LE. “Seperable whorl-specific expression and negetive regulation by enhancer elements within the *agamous* second intron.”. *Plant Cell*, 12:1799–1810, 2000.
- [116] Czech MP. “PIP2 and PIP3: complex roles at the cell surface”. *Cell*, 100:603–606, 2000.
- [117] Chabrier-Rivier N, Fages F, and Soliman S. “The biochemical abstract machine biocham”. *Lecture Notes in Bioinformatics*, 3082:172–191, 2005.
- [118] Campbell NA, Reece JB, Taylor MR, Simon EJ, and Dickey JL. *Biology: Concepts and Connections, 6/E*. Benjamin Cummings, 2009.
- [119] Thieffry D Naldi A and Chaouiya C. “Decision diagrams for the representation and analysis of logical models of genetic networks”. *Springer LNCS/LNBI*, 4695:233–247, 2007.
- [120] Roig O, Cortadella J, and Pastor E. “Verification of asynchronous circuits by BDD-based model checking of Petri nets”. *Lecture Notes in Computer Science, Springer Berlin/Heidelberg*, 935:374–391, 1995.
- [121] Blume-Jensen P and Hunter T. “Oncogenic kinase signalling”. *Nature*, 411:355–365, 2001.
- [122] Cohen P and Frame S. “The renaissance of GSK3”. *Nature Reviews Molecular Cell Biology*, 2:769–776, 2001.
- [123] Jiang P, Du W, Heese K, and Wu M. “The Bad guy cooperates with a good cop p53: Bad is transcriptionally up-regulated by p53 and forms Bad/p53 complex at the mitochondria to induce apoptosis.”. *Molecular and Cell Biology*, 26:9071–9082, 2006.
- [124] Jiang P, Du W, and Wu M. “p53 and Bad: remote strangers become close friends”. *Cell Research*, 17:283–285, 2007.
- [125] Kahlem P and Birney E. “Dry work in a wet world: computation in systems biology”. *Molecular Systems Biology*, 2, 2006.
- [126] Crino PB, Nathanson KL, and Henske EP. “The Tuberous Sclerosis Complex”. *The New England Journal of Medicine*, 355:1345–1356, 2006.



- [127] Maisonpierre PC, Suri C, Jones PF, Bartunkova S, Wiegand SJ, Radziejewski C, Compton D, McClain J, Aldrich TH, Papadopoulos N, Daly TJ, Davis S, Sato TN, and Yancopoulos GD. “Angiopoietin-2, a natural antagonist for Tie2 that disrupts in vivo angiogenesis.”. *Science*, 277:48–50, 1997.
- [128] Burnett PE, Barrow RK, Cohen NA, Snyder SH, and Sabatini DM. “RAFT1 phosphorylation of the translational regulators p70 S6 kinase and 4E-BP1”. *Proceedings of National Academy of Science of the USA*, 95:1432–1437, 1998.
- [129] Hofstadt R and Thelen S. “Quantitative modeling of biochemical networks”. In *Silico Biology 1*, pages 39–53, 1998.
- [130] Losick R and Desplan C. “Stochasticity and cell fate”. *Science*, 320: 65–68, 2008.
- [131] Thomas R. “Regulatory networks seen as asynchronous automata: a logical description.”. *Journal of Theoretical Biology*, 153:1–23, 1991.
- [132] Thomas R, Thieffry D, and Kaufman M. “Dynamical behaviour of biological regulatory networks-i. biological role of feedback loops and practical use of the concept of the loop-characteristic state.”. *Bulletin of Mathematical Biology*, 57:247–276, 1995.
- [133] Bryant RE. “Graph-Based Algorithms for Boolean Function Manipulation”. *IEEE Transactions on Computers*, 35:677–691, 1986.
- [134] Kim RH and Mak TW. “Tumours and tremors: how PTEN regulation underlies both”. *British Journal of Cancer*, 94:620–624, 2006.
- [135] Bahar RI, Frohm EA, Gaona CM, Hachtel GD, Macii E, Pardo A, and Somenzi F. “Algebraic decision diagrams and their applications”. *Proceedings of the 1993 IEEE/ACM international conference on Computer-aided design, Santa Clara, USA*, pages 188–191, 1993.
- [136] Garofalo RS, Orena SJ, Rafidi K, Torchia AJ, Stock JL, Hildebrandt AL, Coskran T, Black SC, Brees DJ, Wicks JR, McNeish JD, and Coleman KG. “Severe diabetes, age-dependent loss of adipose tissue, and mild growth deficiency in mice lacking Akt2/PKB $\beta$ ”. *Journal of Clinical Investigation*, 112:197–208, 2003.
- [137] Yeung RS. “Multiple roles of the tuberous sclerosis complex genes”. *Genes Chromosomes Cancer*, 38:368–375, 2003.
- [138] Fields S and Song O. “A novel genetic system to detect protein-protein interactions”. *Nature*, 340:245–246, 1989.

- [139] Huang S, Eichler G, Bar-Yam Y, and Ingber DE. “Cell fates as high-dimensional attractor states of a complex gene regulatory network”. *Physics. Review Letters*, 94:128701:1–128701:4, 2005.
- [140] Klamt S, Saez-Rodriguez J, Lindquist JA, Simeoni L, and Gilles ED. “A methodology for the structural and functional analysis of signaling and regulatory networks”. *BMC Bioinformatics*, 7, 2006.
- [141] Li S. “A quantitative study of the division cycle of caulobacter crescentus stalked cells”. *PLoS Computational Biology*, 4, 2008.
- [142] Velasco-Miguel S, Buckbinder L, Jean P, Gelbert L, Talbott R, Laidlaw J, Seizinger B, and Kley N. “PA26, a novel target of the p53 tumor suppressor and member of the GADD family of DNA damage and growth arrest inducible genes”. *Oncogene*, 18:127–137, 1999.
- [143] Kauffman SA. “Metabolic stability and epigenesis in randomly constructed genetic nets.”. *Journal of Theoretical Biology*, 22:437–467, 1969.
- [144] Szabo SJ, Sullivan BM, Peng SL, and Glimcher LH. “Molecular mechanisms regulating Th1 immune responses”. *Annual Reviews Immunology*, 21:713–758, 2003.
- [145] Zhang SQ, Ching WK, Ng MK, and Akutsu T. “Simulation study in probabilistic Boolean network models for genetic regulatory networks”. *International Journal of Data Mining and Bioinformatics*, 1:217–240, 2007.
- [146] Datta SR, Dudek H, Tao X, Masters S, Fu H, Gotoh Y, and Greenberg ME. “Akt phosphorylation of BAD couples survival signals to the cell-intrinsic death machinery”. *Cell*, 91:231–241, 1997.
- [147] Dever TE. “Gene-specific regulation by general translation factors”. *Cell*, 108:545–556, 2002.
- [148] Devloo V, Hansen P, and Labb M. “Identification of all steady states in large biological systems by logical analysis”. *Bulletin of Mathematical Biology*, 65:1025–1051, 2003.
- [149] Stambolic V, Ruel L, and Woodgett JR. “Lithium inhibits Glycogen Synthase Kinase-3 activity and mimics wingless signalling in intact cells”. *Current Biology*, 6:1664–1668, 1996.
- [150] van Slegtenhorst M, de Hoogt R, Hermans C, Nellist M, Janssen B, Verhoef S, Lindhout D, van den Ouweland A, Halley D, Young J, Burley M, Jeremiah S, Woodward K, Nahmias J, Fox M, Ekong R, Osborne J, Wolfe J, Povey S, Snell RG, Cheadle JP, Jones AC, Tachataki M, Ravine D, Sampson JR, Reeve MP, Richardson P, Wilmer F, Munro C, Hawkins TL, Sepp T, Ali JB, Ward S, Green AJ, Yates JR, Kwiatkowska

- J, Henske EP, Short MP, Haines JH, Jozwiak S, and Kwiatkowski DJ. "Identification of the tuberous sclerosis gene TSC1 on chromosome 9q34". *Science*, 277:805–808, 1997.
- [151] Reddy VN, Liebman MN, and Mavrovouniotis ML. "Qualitative analysis of bio-chemical reaction systems". *Computational Biology and Medical Informatics*, 26:9–24, 1996.
- [152] Bai X, Ma D, Liu A, Shen X, Wang QJ, Liu Y, and Jiang Y. "Rheb activates mTOR by antagonizing its endogenous inhibitor, FKBP38". *Science*, 318:977–980, 2007.
- [153] Chen Y, Rodrik V, and Foster DA. "Alternative phospholipase D/mTOR survival signal in human breast cancer cells". *Oncogene*, 24:672–679, 2005.
- [154] Sun Y, Fang Y, Yoon MS, Zhang C, Roccio M, Zwartkruis FJ, Armstrong M, Brown HA, and Chen J. "Phospholipase D1 is an effector of rheb in the mTOR pathway". *Proceedings of National Academy of Science of the USA*, 105:8286–8291, 2008.
- [155] Kohavi Z. *Switching and Finite automata theory*. Mc Graw-Hill, 1970.
- [156] Liu Z and Meyerowitz EM. "*LEUNIG* regulates *agamous* expression in *arabidopsis* flowers". *Development*, 121:975–991, 1995.



# Curriculum Vitae

---

## Abhishek GARG

---

### Education

---

- 2005 – 2009(exp.) **PhD Candidate** Computer Science  
Ecole Polytechnique Fédérale de Lausanne,  
Lausanne, Switzerland.
- 2000 – 2004 **Bachelors of Engineering** Information Technology  
Indian Institute of Information Technology, WBUT,  
Calcutta, India.

---

### Publications

---

#### Journal papers

- [J1] Garg A, Mohanram K, Di Cara A, De Micheli G and Xenarios I. “Modeling stochasticity and robustness in gene regulatory networks”. *Bioinformatics*, 25:i101-i109, 2009.
- [J2] Garg A, Di Cara A, Mendoza L, Xenarios I and De Micheli G. “Synchronous vs. Asynchronous modeling of gene regulatory networks”. *Bioinformatics*, 24:1917-1925, 2008.
- [J3] Di Cara A, Garg A, De Micheli G, Xenarios I and Mendoza L. “Dynamic simulation of regulatory networks using SQUAD”. *BMC Bioinformatics*, 8:462, 2007.

#### Conference papers

- [C1] Garg A, Banerjee D and De Micheli G. “Implicit methods for probabilistic modeling of gene regulatory networks”. *30th IEEE EMBS Annual International Conference 2008, Vancouver, Canada*.

- [C2] Garg A, Mendoza L, Xenarios I and De Micheli G. “Modeling of multiple valued gene regulatory networks”. *29th IEEE EMBS Annual International Conference 2007, Lyon, France.*
  - [C3] Garg A, Xenarios I, Mendoza L and De Micheli G. “An efficient method for dynamic analysis of gene regulatory networks and in-silico gene perturbation experiments”. *11th Annual International Conference on Research in Computational Molecular Biology (RECOMB) 2007, San Francisco, USA.*
  - [C4] Yoon S, Garg A, Chung E-Y, Park HS, Park WY, De Micheli G. “Exploiting binary abstractions in deciphering gene interactions”. *28th IEEE EMBS Annual International Conference 2006, New York, USA.*
  - [C5] Garg A, Gharpurey R, Charbon E. “Reduced order models of integrated RF spiral inductors with geometrical and technological automatic parameterization”. *IEEE BMAS conference 2005, San Jose, USA.*
-