AUTOMATIC RECONSTRUCTION OF GENE REGULATORY NETWORK MODELS FROM TIME SERIES DATA

By

Aastha

2009EE50052

Submitted

In partial fulfillment of the requirements of the degree of

Dual Degree in Electrical Engineering

Guidance of

Prof Sumeet Agarwal and Prof Ashwin Srinivasan

Department of Electrical Engineering

Indian Institute of Technology, Delhi

20 May, 2014

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

G	Networks
N	Number of species / places in a petri net/in a LGTS
T	Transitions in the PN/LGTS
A	Arcs between places and transitions
W	Weights of arcs i.e. stoichiometric coefficient in reaction
r	Reaction vector
fr	Boolean control function

SHORT FORMS AND ACRONYMS

AI	Artificial Intelligence
FSM	Finite State Machine
GRN	Gene Regulatory Networks
LGTS	Logic Guarded Transition System
ML	Machine Learning
PN	Petri-Net
LUT	Look-up table

ABSTRACT

Network biology and systems biology offer a novel way of approaching drug discovery. The models that are developed consider not only the chemistry of molecules but also the environment of protein targets, and the effects of modifying them [24]. The problem of modelling the biological networks is very challenging but equally important for drug design and delivery. In contrast to the reductionist molecular biology, systems biology's holism employs computational modelling, theory and wet-lab experiments jointly, to model biological systems as networks. In order to develop efficacious drugs, accurate modelling of these systems is very important.

In the first part of the project we implemented a system which can infer extended PN models from the time series data of the system. The system finds all PN conformal with the time series data given as input to the system. Rather than using the conventional Quine McCluskey algorithm to solve for control arcs (suggested by Durzinsky et al[3]), we have proposed an approximate way which is more efficient, and better considering the large no of species present in a typical GRN. In the second leg of the project we implemented a system that infers LGTS models of the system from the time series data. The extended PN and LGTS models are Turing equivalent. However the guards of LGTS are more flexible and expressive and therefore make LGTS a richer model to represent biological networks. We have successfully tested both the implementations of the system. In both the cases we found that the system could successfully reconstruct the PN which was used in the first place to obtain the time series data. The MATLAB implementation reconstructs all possible networks which are conformal with the data input, while the LGTS completely describes the minimal network which is conformal with the input data.

This work was focused on developing an efficient working implementation of both these modelling techniques and comparing their strengths, space-time complexity. The results of both these implementations are in a good match.

CERTIFICATE

I certify that this report explains the work carried out by me in the Course EED851 – Major Project Part 1 and EED852 – Major Project Part 2 under the supervision of **Dr. Sumeet Agarwal and Dr Ashwin Srinivasan**. The contents of the report including text, figures, tables, Matlab simulations etc. have not been reproduced from other sources such as books, journals, reports, manuals, websites, etc. Wherever limited reproduction from another source had been made the source had been duly acknowledged at that point and also listed in the References.

Aastha

2009EE50052

(Indian Institute of Technology, Delhi) Date :

CERTIFICATE

This is to certify that the dissertation titled ‘Automatic Reconstruction of Petri-net models of Gene Regulatory Network from time series data’ being submitted by Aastha, Entry number 2009EE50052, in partial fulfilment of the requirements for the award of the Dual degree in Electrical Engineering, Department of Electrical Engineering, Indian Institute of Technology , Delhi, is a bonafide record of the work carried out by her in the course EED851 – Major Project Part 1 and EED852 – Major Project Part 2 under my supervision. The matter submitted in this dissertation has not been admitted for an award of any other degree anywhere.

Dr. Sumeet Agarwal,
Assistant Professor,
Indian Institute of Technology,
New Delhi

Dr Ashwin Srinvasan
Professor,
IIIT , New Delhi

ACKNOWLEDGEMENT

It is my pleasure to thank Dr. Sumeet Agarwal and Dr Ashwin Srinivasan for giving me the opportunity to work on such an interesting topic under their guidance. I am grateful to them for their valuable time and suggestions without which I wouldn’t have succeeded in my endeavor. I would also like to thank the members of the evaluation committee for regularly reviewing my work and giving me positive feedback and guiding me in the right direction. I am grateful to all the professors of electrical engineering department who have taught me.

I would also like to extend my gratitude to the Multimedia lab staff and the PhD students working in the lab who were always there to help me whenever I needed them. I also want to thank my friends and family for their constant support and cooperation throughout the course of the project.

Aastha

2009EE50052

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

In the current scenario health, medicine and drug delivery are an important area of research. Models of normal, disease perturbed and drug affected regulatory networks reduce the time & cost of new drug development and are thus very effective in drug design and discovery [20] [21].

However, the problem of inference of these biological networks is quite complicated. The state of the art algorithms in this area are deficient and there is a vast scope of finding new, efficient algorithms that help in inferring these networks. Also, there a lot of mathematical models which represent these with varying degrees of detail. Finding richer, accurate and more expressive models which are simple to model mathematically is another exciting research avenue.

1.2 PROBLEM DEFINITION

The Problem of automatic network reconstruction

In systems biology/medicine considerable effort goes into figuring out what components in the reaction mixture interact with which others in what sequence (reaction mechanisms) and how does the system evolve. Finding which reactants/intermediates in particular catalyze/inhibit the production of the other intermediates is important in some sense to reconstruct the network model. Traditional approach to find these reaction mechanisms is to use biological knowledge and iteratively refine it with the experimental data and build a model. The model obtained can be simulated to test performance but it cannot be said if alternative models exist which could possibly explain the model better. There arises a concept of best fitting model, minimal model etc in the set of plausible networks.

Here in this work we try to construct the PN models that can explain the input time series data and explain the interactions in the system. Also, these networks serve to predict system response under various conditions and the time series evolution of the system. We try to develop techniques that can automate this work and make the iteration faster. Also we expect that these methods list accurate models but at the same time these are not too computationally intensive & have realistic execution time.

1.3 SYSTEMS BIOLOGY

Systems Biology *is a holistic approach towards the study of interactions between the components of a biological system, and how these interactions give rise to the function and behavior of that system* [17]. Systems biology uses mathematical and computational models to model metabolic and signaling networks. "*Systems biology has integrated life science disciplines with mathematics, engineering and computer science to address complex problems in human biology and medicine"* [22].

1.4 GENE REGULATORY NETWORKS

Gene is a hereditary unit mainly consisting of DNA & RNA. It has the necessary information to build/maintain organism's cells. It is also responsible to pass on one's genetic traits to one's off-springs. **Gene Expression** is a process by which information from a gene is used in synthesis of proteins and biological chemicals. All the organisms from unicellular to the most complex of beings employ this process. The mRNA and various types of proteins present in the cell arise from gene expression. These chemicals together with their complex interactions comprise the gene regulatory network.

A gene regulatory network or a genetic regulatory network (GRN) regulation of expression levels of mRNA and proteins in a cell by other chemicals like DNA [19].

Gene expression in a living being generates mRNAs, proteins, and biological chemicals. These interact with each other in the cell, or with the environment or with the surrounding and far away cells by mediating signals. A GRN includes these biological molecules together with their interactions. In this network, the nodes depict the species involved. Inductive & inhibitory interactions, molecular reactions etc surface as edges between these nodes.
System models and network representation represent the system's chemical dynamics, and the ways in which one species can possibly affect others directly or indirectly. [19].

1.5 PETRI-NETS AND THEIR TYPES

Petri nets are also called P/T nets or place transition nets. PN are a standard mathematical modelling language. It can be seen as a bipartite graph (The places being the red nodes and the transitions being the blue nodes). Directed arcs between the places and transitions help to determine the pre & post conditions of transitions firing. PN are very convenient to describe reactions, networks & interactions.

Petri Nets have been chosen over other modelling techniques because –

- Interactions can be easily modeled
- concurrency issues are handled suitably
- transitions rates map kinetic rates of chemical reactions
- graphical description of system aids understanding

When read and inhibitory arcs are present in a petri net it is called an **extended petri net**. The PN are '*simple*' that is there are no bi-directed arcs in the network i.e. there is no place and transition connected in 2 ways.

1.6 LGTS – LOGIC GUARDED TRANSITION SYSTEM

We know that concurrency and stochasticity of petri-nets makes them most suitable mathematical model to represent biological networks. Pure and extended PNs can be represented as a special case of LGTS. An LGTS can be understood as an extended PN but with guards rather than control arcs on the transitions. An LGTS is more expressive and hence is a more powerful way of modelling systems because guards help to impose logical as well as linear constraints on the firing of a transition. The system is ideal for quantitative and qualitative modelling and analysis of the system.

1.6.1 FSM

- It is a computing machine which describes
- state of the system at an instant.
- states the system could transition to when presented with a certain inputs
- outputs that the system generates when it makes one of the possible transitions

A system can be represented using a FSM only if

- only a finite number of states (as suggested by the name) can be achieved
- set of input events at any state are finite
- set of reachable states from any given state are finite
- set of outputs that it can generate are finite

FSM is one of the most common and traditional ways of depicting systems.

The entire history of inputs gets captured in the form of current state. In other words, the only thing that determines the next state is the current state and the input in the current state. The history of inputs has no effect (except in determining the present state of the system). Each state can be thought of as having a look up table (LUT). On the arrival of an input it just checks the LUT and finds the next state.

Vending machines, elevators, compilers, software architecture are often described using FSM.

A FSM can be used in many ways including acceptors, classifiers, transducers and sequences. In an acceptor sort of an implementation start and end are defined to be two of the machine's "special" states. FSM can be used to find what sequence of symbols can be used to move it from the start to the final state or generally between a pair of states. Thus FSM helps us to find acceptable sequences.

1.7 SNOOPY

In the present work we have used Snoopy to simulate extended stochastic petri nets. The reconstructed networks are also graphically depicted using the same tool. It is an extensible, platform independent, easy to use tool for depicting, modelling and simulating petri nets.

More about it can be found in the report Petri nets in Snoopy: A unifying framework for the graphical display, computational modelling, and simulation of bacterial regulatory networks [18].

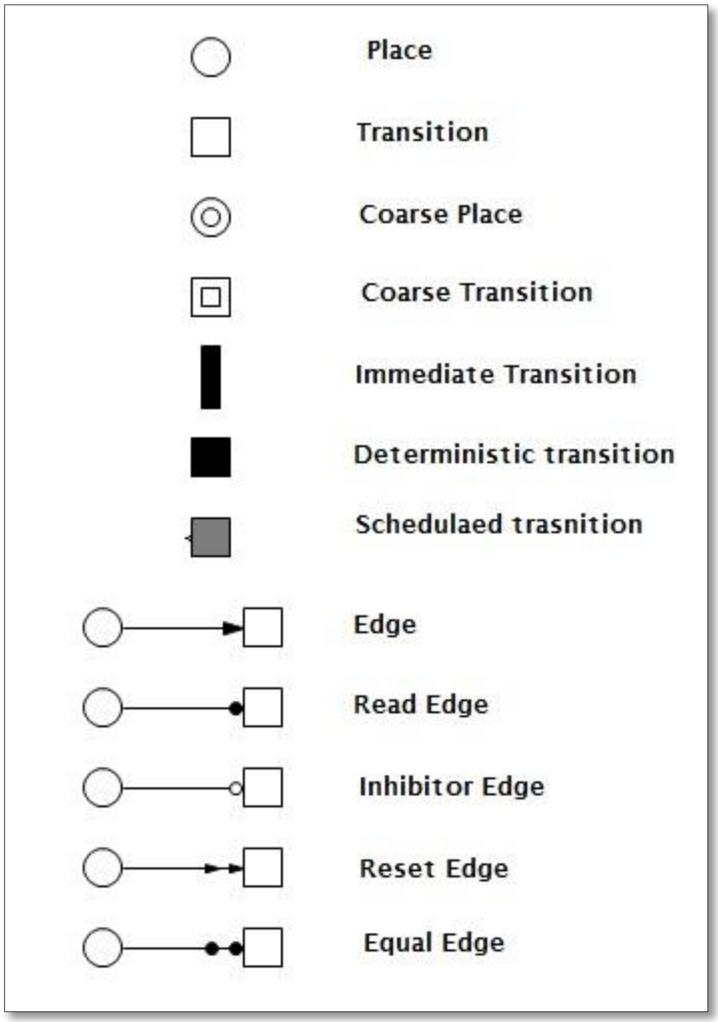


Fig 1.1 – Petri net elements and their graphical representation in snoopy

1.8 LGTS vs. PN

The expressive power of a PN and that of a LGTS is same (they are known to be Turing Equivalent). Any extended Petri Net can be depicted using a LGTS system. In a LGTS instead of the read/inhibit arcs we have guards (linear and logical constraints in predicate form). If the guard in the constraint box evaluates to true then the transition fires and tokens are transferred as usual. It lists a conjunction of the pre, post conditions and the invariants related to the transition.

$$\textbf{Guard}_t = \textbf{Pre Conditon}_t \wedge \textbf{Post Conditions}_t \wedge \textbf{Invariants}_t$$

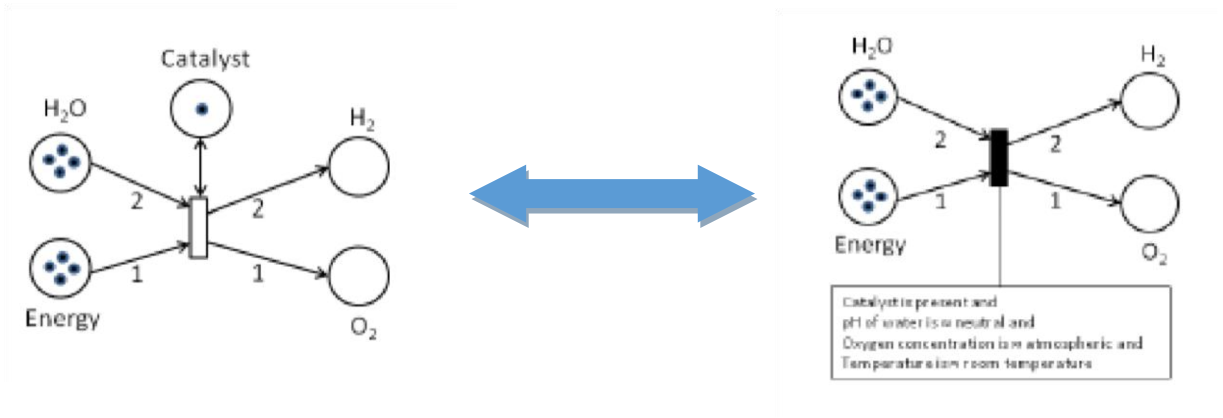

Catalyst
H_2O
H_2
2
2
1
1
Energy
O_2
H_2O
H_2
2
2
1
1
Energy
O_2
Catalyst is present and
pH of water is ≈ neutral and
Oxygen concentration is ≈ atmospheric and
Temperature is ≈ room temperature

Fig 1.2 The extended PN and the LGTS system are essentially equivalent. They have same Turing degree.

The experimental data in the form of concentrations of reactants is converted in the form of predicates i.e. the time series data is in predicate form or logical form. These serve as positive and negative examples. These +ve and –ve examples alongwith background data are fed to the ILP engine which together with the help of a constraint solver generates a network model .

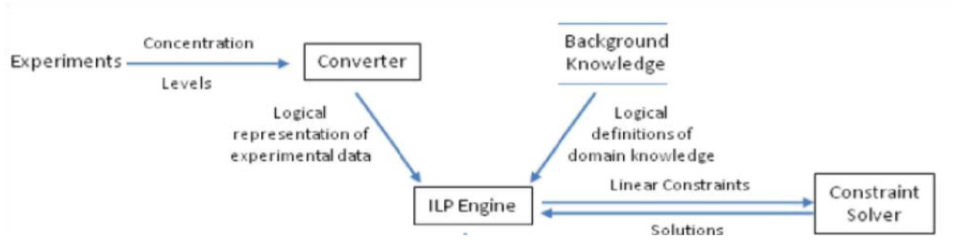


Fig 1.3 The basic setup of the LGTS system

1.9 FLOW OF WORK – Major project part 1 + part 2

1) Literature reviewed to understand the basics of Petri Nets, Gene Regulatory Networks, LGTS and biological networks.

2) Study and understanding of various constraints and implementation of the same in the form of combinatorial techniques to help in inferring Logic Guarded Transition Systems and Petri Nets.
3) Implementation of 'Automatic Network Reconstruction Algorithm' with a special focus on inferring the control arcs from the time series data of the system.
4) Testing the implementation to infer Phosphate regulatory network in enteric bacteria.
5) Development and Implementation of a LGTS inferring system
6) Testing and benchmarking the LGTS inferring system against the Automatic Network Reconstruction Algorithm on the Phosphate regulatory network in enteric bacteria.

1.10 ORGANIZATION OF THE REPORT

This report contains the following sections: Chapter 1 gives a basic introduction to the area of system biology and the problem we are delving with in this work. Chapter 2 contains the literature review. Chapter 3 discusses the biological system we have been working with and the pseudo codes of the systems implemented in this work. We have succinctly presented the results of the two implementations in Chapter 4. . The conclusions and summary of project is given in Chapter 5. Appendix discusses the theoretical differences between a PN and a FSM, and how to interconvert the two.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

The available literature has been studied to understand the following aspects:

1. Basics of gene-regulatory network and their modelling,
2. Literature on Petri-nets, their types and representation
   Pure & extended petri-net inference, efficiently solving large Boolean maps
3. First order logic, Inductive Logic Programming (PROLOG), Logic guarded transition system

Network diagnostics have been used to investigate networks in [1]. Relating network structure to network functions is another direction of effort. For example *discovering evolutionarily significant aspects of metabolic networks, detecting structural constraints on particular network types, and constructing summary statistics for efficient model fitting to networks. It is a data-driven approach to aid in the understanding of networked systems* [1].

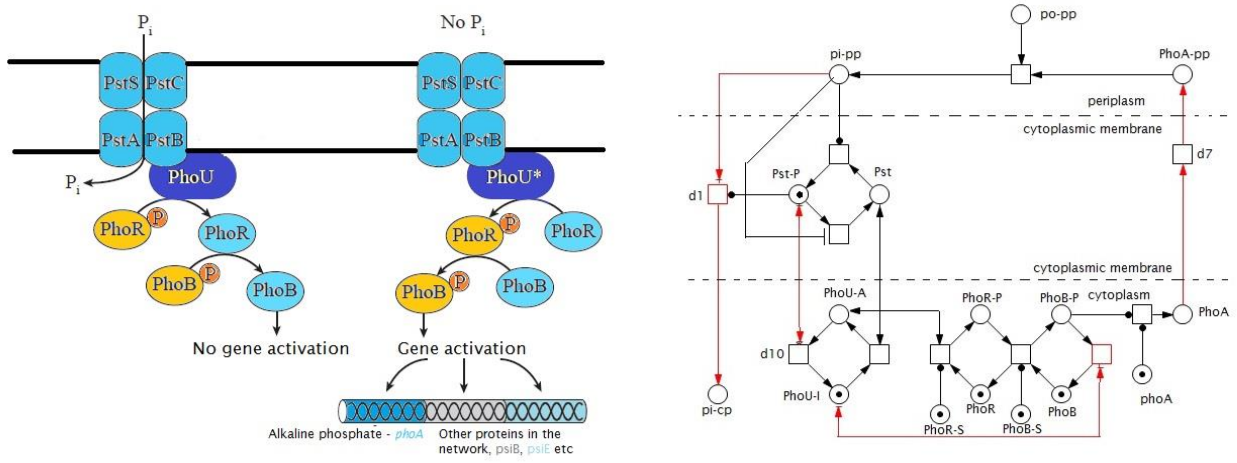
2.2 MODELLING GENE REGULATORY NETWORK

Boolean networks, logic circuits, probabilistic Boolean networks, Ordinary differential equations (ODEs), stochastic ODEs, qualitative differential equations and more have been used in literature to model GRNs. A relationship between two most popular modelling techniques probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks is given in [39].

The paper "learning petri net models of Biological systems using ILP by Ashwin Srinivasan and Michael Bain" [40] suggests that using ILP system for learning petri nets is one efficient way of approaching the modelling problem when using combinatorial techniques over the search space. Paul nurse [41] suggests that modelling attempts are futile unless we understand how information is managed in living systems and how this brings about higher-level biological phenomena. He also stresses on the need of an appropriate language or model to describe this information processing in biological systems. He proposes logic circuits like interpretation of biological phenomena and interactions. This is a 3 phase task involving - firstly, breaking down the logic circuits in cells into logic modules like feedback-loops & therefore constructing a logic 'tool kit. Knowledge of which modules are operational and how these are linked into circuits will help us to understand the flow of information. Next phase being analyzing biochemistry and linking it to these logic modules. The last phase being mapping of cellular chemistry and the logic tool kits. Hence an unusual 'scale-free' or complex network/closed circuit can be built where the linkages/edges will be heterogeneous (some will represent physical interactions & others chemical transformations). Narai et al. [42] suggests another probabilistic approach towards estimating the gene network

2.3 PETRI-NETS AND THEIR TYPES

Petri-nets were first introduced by Carl Adam Petri in 1939 to model chemical processes [29].



2.1 Control of a metabolic pathway and its Petri net representation

A pure petri net is a simple model with places, transitions and directed arcs between them. Places have infinite capacity to hold tokens. A transition fires when it is enabled. In a simple petri net this just needs the pre-places of the transition to have the required number of tokens. Default multiplicity of an arc is unity. Other than that is specified. The direction of the arcs tell us about the direction of flow of tokens when a transition fires. Reset arcs also exist in literature.

Carl Adam Petri [29] also gives the basic principles of working of Petri Nets. It explains the versatility of PNs and how they can be used to model various systems that involve intense concepts like concurrency and synchronization. Sequence modelling is trivial and can be achieved using a plethora of models. PNs can also model conflict. Confusion which is the fusion of conflict and concurrency can also be modelled using PNs. Another celebrated topic of Petri-Nets is their reachability graph. Multi-Terminal Binary Decision Diagram (MTBDD), generalized version of BDD, used for verification of probabilistic systems is given in [30].

The concept and working of coloured petri nets [31, 32], continuous petri nets [34], differential petri nets, hybrid petri nets[34], timed petri nets, stochastic petri-nets, hierarchical petri-nets, extended petri-nets, prioritized petri-nets, music petri net, dualistic petri nets and more are interesting topics.

2.4 FIRST ORDER LOGIC, ILP & PROLOG

In the book Prolog Programming for artificial intelligence by Ivan Bratko, the chapter on Inductive Logic Programming gives a precise definition to Inductive Logic Programming. Inductive Logic Programming, as the name suggests, comes under the logic programming paradigm. At the same time it is influenced by Machine Learning. It is an approach to learning where we express the learnt hypothesis using predicate logic. ILP uses positive examples, negative examples and background knowledge to construct a hypothesis. Necessity and sufficiency requirements ensure that the hypothesis is not generalized beyond need but can explain all positive examples. Consistency ensures that no negative examples or background information get violated.

Aleph stands for ***A** **L**earning **E**ngine for **P**roposing **H**ypotheses*. Aleph is de facto best Inductive Logic Programming (ILP) system [48].

Prolog is a general purpose logic programming language associated with artificial intelligence and computational linguistics [48].

2.5 LOGIC GUARDED TRANSITION SYSTEM

A novel approach to modelling has been proposed in this project. This modelling approach involves employing a core of FSM with linear and logical constraints acting as guards for the FSM transitions. This has been proposed by Ashwin Srinivasan, Michael Bain and K Sriram and is currently under development.

2.6 SOLVING LARGE BOOLEAN MAPS

Espresso logic minimizer [44] developed by at IBM by Robert Brayton, and published and hosted by Richard Rudell, the center for electronic system design, Berkeley, is a well-known multi-valued PLA minimization tool and is often used for simplifying electronic gate circuits [45].

Another elementary and celebrated algorithm that is used in Boolean function minimization is the method of prime implicants, better known as the Quine-McCluskey algorithm. This algorithm has an exponential time complexity with respect to number of variables in the Boolean map.

The problem in this thesis was a little different from that of minimizing a karnaugh map to obtain a Boolean function. Here we trying to figure out the minimal solutions that can explain all the observations. An approximate version of Quine McCluskey was implemented to achieve the same. If **m** is the order of **number of vectors in the state matrix**, **n** is the **number of prime implicants**, then the algorithm takes **$O(n*m)$** to find the possible single variable prime implicants , while to figure out the pairs of prime implicants the time taken is of the **$O(n^2 * m)$**. However, the latter is found only if no single variable terms could be found. In general to find a term of k variables the time complexity is of the order **$O({}^{n}C_{k} * m)$.**

This chapter presented an account of literature review carried out during the course of this project. The next chapter is a terse description of the line of thought and skeletons of most implementations done in the project.

CHAPTER 3

SYSTEM DESCRIPTION

3.1 PHOSPHATE REGULATORY NETWORK IN ENTERIC BACTERIA

In DNA of viruses and bacteria, structural genes coding for functionally related proteins (say the proteins involved in a particular metabolic pathway or signaling pathway) form clusters/groups. Thus in the DNA there are groups of functionally related proteins. Such a group is called an ***operon***. This helps to control protein synthesis in coordination with the cell needs. It enables the cell to produce proteins only when needed. This cluster of functionally related genes is controlled by a single promoter.

The expression of genes in the *Pho operon* are controlled by a network of phosphate sensing and signal transducing proteins called the phosphate regulatory network. The genes in pho operon are responsible for metabolism of phosphate in enteric bacteria.

The petri net model of this network was used as a test case to test the 2 implementations of inferring models of gene regulatory networks.

3.2 BIOLOGY OF PHOSPHATE REGULATION

A microorganism requires inorganic phosphate to synthesize DNA, RNA. PstSCAB complex takes up inorganic phosphate and transports it. When there is a deficiency of phosphate, the substrate for the phosphate transporter gets deficient. This triggers transcription of phoA gene which encodes alkaline phosphate (through proteins like PhoU, PhoR, PhoB). This alkaline phosphate produces inorganic phosphate by degrading any organic phosphate present in the cell, which is immediately available to PstSCAB complex. However, this entire process is very energy intensive. So, there is a negative feedback loop, through which the inorganic phosphate in the cell controls transcription of the phoA gene.The authors [3] claim that all these interactions (including the negative feedback loop) have been successfully captured in the petri net model of the system. Snoopy was used to build, simulate and execute these petri-nets.

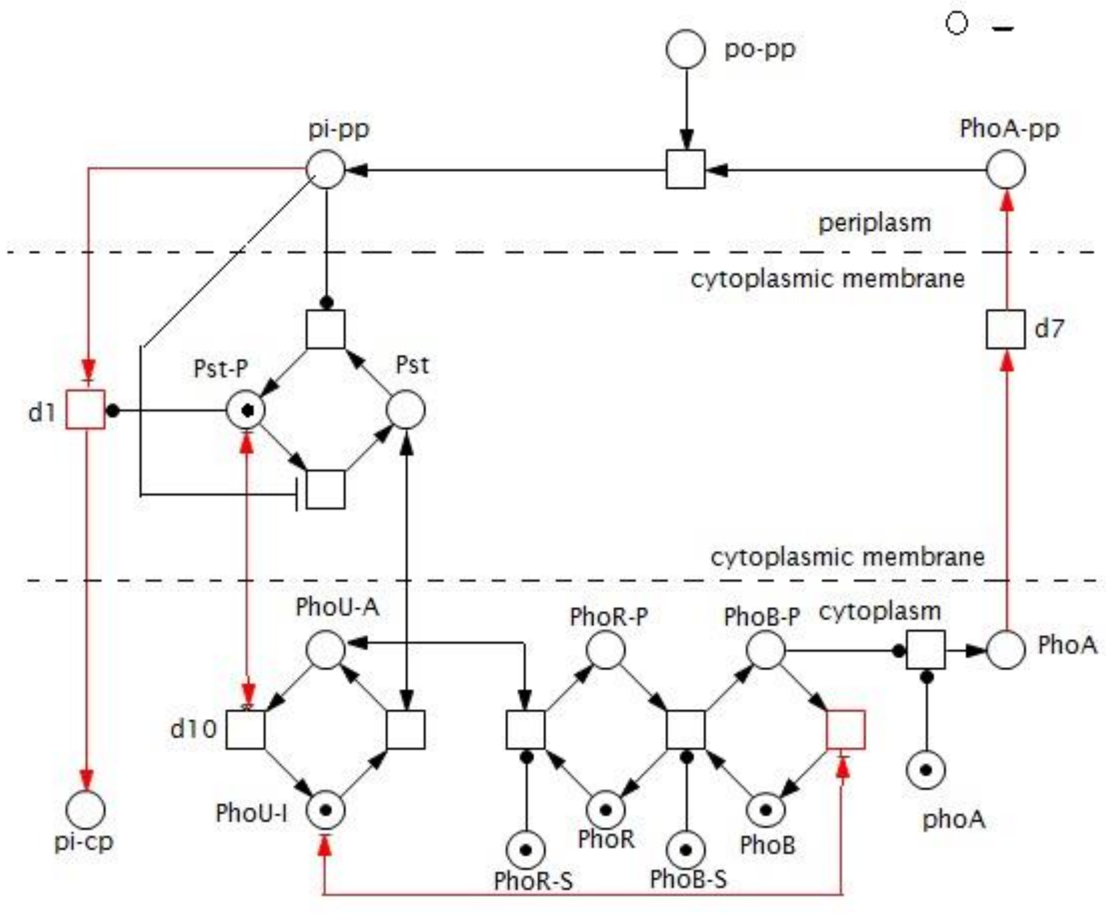


Fig 3.1 Petri net representation of phosphorous regulation cycle

TIME SERIES DATA OF THE PHOSPHORYLATION NETWORK

This time series data is same as that used by durzinky et al in [3].

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
4	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	0
5	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	0	1	1
6	1	1	1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0

7	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	0	0	1
8	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1
9	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
12	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
16	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	0	0
												x									**T**			

Table 3.1 (a) Time series data of the phosphorylation network

	25	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**
1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0
5	1	1	1	0	0	0	0	1	0	1	1	0	1	1	1	0	1	1	0	1	1	1
6	0	0	0	1	1	1	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
7	1	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	1
8	0	1	0	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	0
9	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1
11	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1

12	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
14	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
			T	**T**	**T**	**T**		**T**							**T**			**T**				**T**

Table 3.1 (b) Time series data of phosphorylation network (contd)

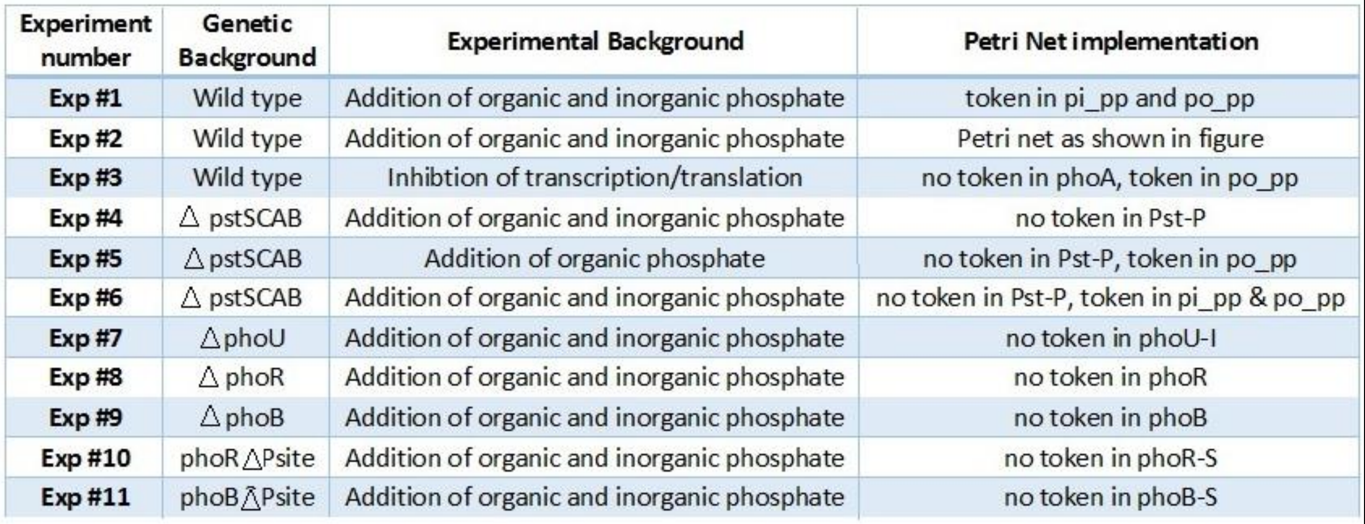
T indicates that this is a terminal vector. The column indices denote the time sequence.

Table 3.2 : Key of places

Places	**Name**
1	Pi-pp
2	Pi-cp
3	Po-pp
4	Pst-p
5	Pst
6	Phou-I
7	Phou-A
8	PhoR
9	PhoR-P
10	PhoR-S
11	PhoB
12	PhoB-P
13	PhoB-S
14	PhoA-T
15	PhoA
16	PhoA-pp

Table 3.3 : Key of transitions

Difference Vector	Input place	Output place
d1	1	2
d2	4	5
d3	6	7
d4	8	9
d5	9,11	8,12
d6		15
d7	15	16
d8	3	1
d9	5	4
d10	7	6
d11	12	11



Experiment number	Genetic Background	Experimental Background	Petri Net implementation
Exp #1	Wild type	Addition of organic and inorganic phosphate	token in pi_pp and po_pp
Exp #2	Wild type	Addition of organic and inorganic phosphate	Petri net as shown in figure
Exp #3	Wild type	Inhibtion of transcription/translation	no token in phoA, token in po_pp
Exp #4	△ pstSCAB	Addition of organic and inorganic phosphate	no token in Pst-P
Exp #5	△ pstSCAB	Addition of organic phosphate	no token in Pst-P, token in po_pp
Exp #6	△ pstSCAB	Addition of organic and inorganic phosphate	no token in Pst-P, token in pi_pp & po_pp
Exp #7	△phoU	Addition of organic and inorganic phosphate	no token in phoU-I
Exp #8	△ phoR	Addition of organic and inorganic phosphate	no token in phoR
Exp #9	△ phoB	Addition of organic and inorganic phosphate	no token in phoB
Exp #10	phoR△Psite	Addition of organic and inorganic phosphate	no token in phoR-S
Exp #11	phoB△Psite	Addition of organic and inorganic phosphate	no token in phoB-S

Table 3.4 - The perturbation caused to the system to obtain time series data. A total of 11 experiments were done to collect data. Reproduced from Durzinsky et al [3].

Abbreviation	Function
PhoA	Alkaline phosphatase enzyme degrading organic phosphate compounds to inorganic phosphate
Pi	Inorganic phosphate
Po	Organic phosphate
PstSCAB	Transmembrane protein complex, transporter of inorganic phosphate
PhoU	Signal transducer relaying the PstSCAB complex activity
PhoR	Phosphorylatable regulatory protein
PhoB	Phosphorylatable regulatory protein
Pi PeriPlasm	Inorganic phosphate in the periplasm
Pi Cytoplasm	Inorganic phosphate in the cytoplasm
Po PeriPlasm	Organic phosphate in the periplasm
PhoA_Periplasam	PhoA protein in the periplasm
PhoA	PhoA protein in the cytoplasm
PhoA mRNA	PhoA gene Mrna
PstSCAB	Transporter of inorganic phosphate, inactive form
PstSCAB-P	Transporter of inorganic phosphate, phosphorylated, active
PhoU_inactive	Signal transducer, inactive form
PhoU_active	Signal transducer, active form
PhoR	PhoR protein, dephosphorylated form
PhoR-P	PhoR protein, phosphorylated form
PhoB	PhoB protein, dephosphorylated form
PhoB-P	PhoB protein, phosphorylated form

TABLE 3.5 Abbreviations and functions Reproduced from Marwan et al [18]

3.3 AUTOMATIC NETWORK RECONSTRUCTION ALGORITHM – DURZINSKY'S METHOD

The networks G = (N, T, A, w)
N is the components involved which transform into the places in the petri net,
T is the interactions between the components which corresponds to transitions in the PN,
A is the arcs between places and transitions, physically representing the flow of mass,
W is the stoichiometric coefficient in the reaction which transforms into weight of an arc.
In an extended petri net a transition is described as a tuple T = (**r, f_r**) , where r is the reaction vector and fr is the control function associated with it. f_r is in the form of a Boolean.

The PN are '*simple*' that is there are no bi-directed arcs in the network i.e. there is no place and transition connected in 2 ways.

3.3.1 DESCRIPTION OF ALGORITHM

Input – Time series data of the system
Output- Extended PN conformal with the input data i.e. Incidence matrix of the petri nets, the control function associated with each transition of petri-net.

Pseudo Code [2] – Reproduced from durzinsky et al [4]

Algorithm: (*Extended network reconstruction*)

```
Input: set of observed components $N$
        monotone experimental data $X', X_T'$
Output: sets of reaction vectors $R$ and functions $f_{\mathbf{r}}$ of catalytic
conformal networks 1: $D := \emptyset$
2: FOR EACH $(\mathbf{x}^j, \mathbf{x}^{j+1}) \in X'$

3: IF $\mathbf{x}^j \in X_T'$ THEN
4: RETURN (no solutions exist)
5: $D := D \cup \{\mathbf{x}^{j+1} - \mathbf{x}^j\}$
6: FOR EACH $\mathbf{d}_j \in D$
7: generate set $R(\mathbf{d}_j) = \mathrm{Box}(\mathbf{d}_j)$
8: generate set $\Lambda(\mathbf{d}_j)$
9: FOR EACH $(\mathbf{x}^j, \mathbf{x}^{j+1}) \in X'$
        10: $\mathbf{d}^j := \mathbf{x}^{j+1} - \mathbf{x}^j$

11: $S_j := \bigcup_{\lambda \in \Lambda(\mathbf{d}^j)}$ 'all permutations of supp$(\lambda)$'
12: IF $\exists \mathbf{y} \in X'_T$: $\mathbf{x}^{j+1} - \mathbf{y} \in \mathrm{Box}(\mathbf{d}^j)$ THEN
13: FOR EACH $s \in S_j$
14: IF $X(s) \cap X'T \neq \emptyset$ THEN
15: $S_j := S_j \setminus \{s\}$
16: FOR EACH $\boldsymbol{\mu} \in [0, |S_j|]_{X'}$
17: $R := \bigcup_{(\mathbf{x}_j, \mathbf{x}_{j+1}) \in X'} R(S_{j,\mu_j})$
18: $F := \emptyset$
19: FOR EACH $\mathbf{r} \in R$
20: generate $f_{\mathbf{r}}$ from $X_T'$ (r) and X' (r) = $\bigcup_j$ X ( S j μj , r) $X(S_{j,\mu_j}, \mathbf{r})$
21: $F := F \cup \{f_{\mathbf{r}}\}$ 22: OUTPUT $(R,
F)$
```

3.3.2 METHODOLOGY

We used the phosphate regulatory network of enteric bacteria to test and benchmark this work. Snoopy[18] can be used to simulate the PN model of this network and time series data of the system can be easily obtained in the form of excel sheets (.csv files). We have used the same time series data as given in [3].

This time series data is given as an input to the system. Another input to the system is a file containing the terminal vectors. The first step is finding the difference vector set by taking difference between the consecutive state vectors (markings) in the time series data. A difference vector can be a result of firing of more than one transition in the petri-net. So these difference vectors can be broken down into sub-reactions called reaction vectors. Also, these sub reactions can occur in any order therefore all possible permutations of occurrence of these sub-reactions are considered. A minimal set of reaction vectors that can explain all the difference vectors obtained so far is found. To find all possible PN structures conformal with the input data all (non-minimal) sets are considered as well.

3.3.3 MONOTONICITY ASSUMPTION

A key point in the algorithm is that we consider monotonicity of data (i.e. the trend of change of tokens between successive states doesn't flip). So the sub-reactions are not arbitrary or erratic fluctuations of concentrations of species and are bounded according to the box rule described above. As the space of sub-reactions is bounded by the box rule, the space where we can find a conformal PN also gets bounded.

Per difference vector one permutation of sub-reactions is chosen to construct the incidence matrix of the extended petri-net. Different permutations of sub-reactions lead to different intermediate states of the system and hence it leads to different control functions and hence different PNs.

3.4 FINDING CONTROL VECTOR

The various system states when a particular reaction fires and the terminal states of the system are taken into account when finding the control function for a reaction vector.

A control function can be formulated that describes under what conditions the sub-reaction fires or more aptly when it doesn't fire. A reaction vector is said to be applicable to a terminal vector if the sum of the 2 vectors yields a valid state vector. In this case we need to we need a control function that prevents the reaction vector from firing at that terminal state. However the control vector should be consistent with the states at which the reaction is known to fire i.e. it shouldn't prevent it from firing at the states where the reaction is known to fire. Such a control function can be obtained by applying Quine McCluskey algorithm to the state vectors where it fired and the terminal vectors where it has fired [3]. In case of the phosphorylation network this problem is same as solving a 256 x 256 karnaugh map or McCluskey for the same. Of this map, very few entries(~ 2 to 10) are known (a combination of 0s and 1s). Rest are all don't cares.

A minimal control function(Boolean) is defined as the one with least number of disjunctions. Also the conjunction on which we perform these disjunctions should involve least number of variables

In this work we have used an approximate version of Quine Mc Cluskey which is more suitable in term of time complexity when the system has a large no of components (the size of the vectors or the size of the K map).

At the end of the iteration the algorithm outputs the incidence matrices and control functions of all the petri-nets conformal with the data.

NOTE : The same algorithm has been used for inferring control arcs for petri-nets and guards in case of LGTS system.

3.4.1 ALGORITHM FOR FINDING CONTROL FUNCTION

If Difference vector = d = $S_j - S_i$ then

All such S_i are cascaded to form a matrix called InitialStateVectors matrix (i.e. every column vector is a state vector in this case).

If Difference vector d + Terminal vector T_i = valid marking, then we say d is applicable to T_i. All such Ti which to which the difference vector d is applicable are included in the set ApplicableTerminalVectors. (A column vector is a terminal state vector)

Across the 11 experiments given in the paper a difference vector may appear multiple times. So, collecting data from all of them and putting it together, difference vector d has multiple InitialStateVectors and ApplicableTerminalVectors . Infact we have 2 matrices corresponding to every difference vectors d – InitialStateVectors and ApplicableTerminalVectors.

Assumption 1: We need to find a control function for a difference vector only if the matrix ApplicableTerminalVectors has one or more vectors. If this matrix has size zero, then we don't need to find a control function for this difference vector. A difference vector which is not applicable to any terminal vector is always on, and has $f_r = 1$.

Every d is a 16 x 1 vector in the phosphoregulatory network of enteric bacteria case. Place(i), where $i \in \{1,16\}$, is the token at the ith place in the vector d.

Algorithm –

If the difference vector needs a control vector then we build one in the following way –

Step 1:

For the difference vector d we fetch – InitialStateVectors and ApplicableTerminalVectors.

Step 2:

If in all the vectors in InitialStateVector matrix, the tokens in **place(i) = 1** while in all the vectors in ApplicableTerminalVectors place(i) = 0 , then place(i) is a catalyst for the reaction represented by d. **Possible control function = read arc from place(i)**

If in all the vectors in InitialStateVector matrix, the tokens in **place(i)= 0** while in all the vectors in ApplicableTerminalVectors place(i) = 1 , then place(i) is an inhibitor for the reaction represented by d. **Possible control function = inhibit arc from place(i)**

Step 3:

If no clear catalyst or inhibitor exist, then we interpret that more than one place is used to build the control function of the reaction i.e. either there are 2 catalysts, or 2 inhibitors or a combination of 1 catalyst and 1 inhibitor for the reaction vector in question.

Step 4:

Here, i ϵ {1,16}, j ϵ {1,16} , j ≠ i.

If in all the vectors in InitialStateVector matrix, the tokens in place(i)**= 1, place(j) = 1** while in all the vectors in ApplicableTerminalVectors (place(i), place(j)) = { (1,0), (0,1), (0,0)} , then place(i , j) are taken as the pair of catalysts for the reaction represented by d. **Possible control function = read arcs from place(i) & place(j).**

If in all the vectors in InitialStateVector matrix, the tokens in place(i)**= 0, place(j) = 0** while in all the vectors in ApplicableTerminalVectors (place(i), place(j)) = { (1,0), (0,1), (1,1)} , then place(i , j) are taken as the pair of inhibitors for the reaction represented by d.

Possible control function = inhibit arcs from place(i) & place(j).

If in all the vectors in InitialStateVector matrix, the tokens in place(i)**= 1, place(j) = 0** while in all the vectors in ApplicableTerminalVectors (place(i), place(j)) = { (1,1), (0,1), (0,1)}, then place(i) is taken as a catalyst while place(j) is taken as a inhibitor for the reaction represented by d.

Possible control function = read arc from place(i) & inhibit arc from place(j).

3.5 PSUEDO CODE FOR FINDING CONTROL FUNCTION

```
Begin function Get_minimal_transition (D1, L1, L2, catalysts, inhibitors, cat_inhibit,
inhibit_cat) :-
Declare Empty_lists : catalyst1, cat_inhibit1, inhibit_cat1, inhibitor1, catalyst2,
cat_inhibit2, inhibit_cat2, inhibitor2.
```

```
V1 = head of L1.
For i = 1 to 16
        x <- tokens in Place(i);
        bool Control_place = 1;
        For vector v in L1 & (v != V1)
            If( tokens in place(i) != x)
            Control_place =0;
            Break;
        end

        If(control_place)
            If(x)  Append(place(i), catalyst1)
            Else     Append(place(i), inhibitors1)
End

V2 = head of L2.
For i = 1 to 16
        x <- tokens in Place(i);
        bool Control_place = 1;
        For vector v in L2 & (v != V2)
            If( tokens in place(i) != x)
                Control_place =0; Break;

            If(control_place)
                If(x)  Append(place(i), inhibitors2)
                Else Append(place(i), catalyst2)
end

For place c1 in catalyst1
        If (c1 is in catalyst2) append (c1, catalyst)
end

For place in1 in inhibitor1
  If (in1 is in inhibitor2) append (in1, inhibitor)
end

if( isEmpty(catalyst) and isEmpty(inhibitor))
    V1 = head of L1.
    For i = 1 to 16
        For j = i+1 to 16
            (X1, X2) <- tokens in Pair(place(i), place(j));
            bool Control_pair = 1;
                For vector v in L1 & (v != V1)
                    If( tokens in Pair(place(i), place(j)) != (x1, x2))
                        Control_pair =0; Break;
                    If(Control_pair)
                        If(x1 = x2 = 1) Append(Pair(place(i), place(j)), catalyst1)
                        Elseif(x1 = 1, x2 = 0) Append(Pair(place(i), place(j)), cat_inhibit1)
                        Elseif(x1 = 0, x2 = 1) Append(Pair(place(i), place(j)), inhibit_cat1)
                        Else Append(Pair(place(i), place(j)),  inhibitors1)
            end
        end
    end

    V2 = head of L2.
    For i = 1 to 16
        For j = i+1 to 16
            (X1, X2) <- tokens in Place(i), Place(j);
            bool Control_pair = 1;
                For vector v in L2 & (v != V2)
                    If( tokens in Pair(place(i), place(j)) != (x1, x2)) Control_pair =0; Break;
                end
                If(Control_pair)
                    If(x1 = x2 = 1)
                        Append(Pair(place(i), place(j))) to  cat_inhibit2, inhibitor2, inhibit_cat2

                    Elseif(x1 = 1, x2 = 0)
                        Append(Pair(place(i), place(j))) to  inhibit_cat2, catalyst2, inhibitor2

                    Elseif(x1 = 0, x2 = 1)
                        Append(Pair(place(i), place(j))) to  cat_inhibit2, catalyst2 , inhibitor2

                    Else
                        Append(Pair(place(i), place(j))) to   inhibit_cat2 , cat_inhibit2 , catalyst2
        end
    end

        For pair_c1 in catalyst1  #[pair_c1 <- Pair(place(i), place(j))]
            If (c1 is in catalyst2) append (c1, catalyst)
        end

        For pair_in1 in inhibitor1  #[pair_in1 <- Pair(place(i), place(j))]
            If (pair_in1 is in inhibitor2)           append (in1, inhibitor)
```

```
                end

                For pair_ic1 in inhibit_cat1 #[pair_ic1 <- Pair(place(i), place(j))]
                        If (pair_ic1 is in inhibit_cat2) append (ic1, inhibit_cat)
                end

                For pair_ci1 in cat_inhibit1
                        If (pair_ci1 is in cat_inhibit2) append (ci1, cat_inhibitor)
                end
        end
end
```

3.6 Bridge

A PN is a generator of infinite state system [15] i.e. the reachability graph of a PN is generally infinite [16]. However, the reachability graph of a k-bounded PN is finite. In other words, the system can reach only a finite number of states when it is a bounded net. The reachable markings depends on the initial marking of the PN. In this work we deal with a binary markings i.e. the number of tokens in a place is either zero or one. When the initial marking of the PN is binary, so are the future markings. Hence, the PN is a safe (k=1) network.

The finiteness of the reachability graph of the PN tells us that the system can be approximated as a Finite state machine. Further, in a LGTS we use guards rather than control arcs which are more flexible and more expressive. Hence, a LGTS is a more accurate/holistic model of a biological system as compared to a Petri-Net.

Both the approaches developed in this thesis work for a safe (1-bounded) networks. Also, the PNs are '*simple*' that is there are no bi-directed arcs in the network i.e. there is no place and transition connected in 2 ways

3.7 FURTHER DETAILS

The code, along with its documentation is available on request. Please write to
ee5090052@iitd.ac.in.
We used the following versions during the development of the code-
PROLOG VERSION: SWI-Prolog version 6.6.1
MATLAB, Version 7.10.0.499 (R2010a), 64-bit (win 64)
The **system** used in all these simulations has a Intel® Core™ i3-3110M CPU @ 2.40GHz with $ GB RAM, 64 bit Windows 8 Operating system installed on it.

In this chapter we described the biological system under consideration, the details and the physical meaning of the simulation experiments carried out to obtain the time series data. Also what is given here is the pseudo-code and a simplified description of the algorithms used to infer networks. In the next chapter we describe the results of reconstruction of these networks from the two implementations.

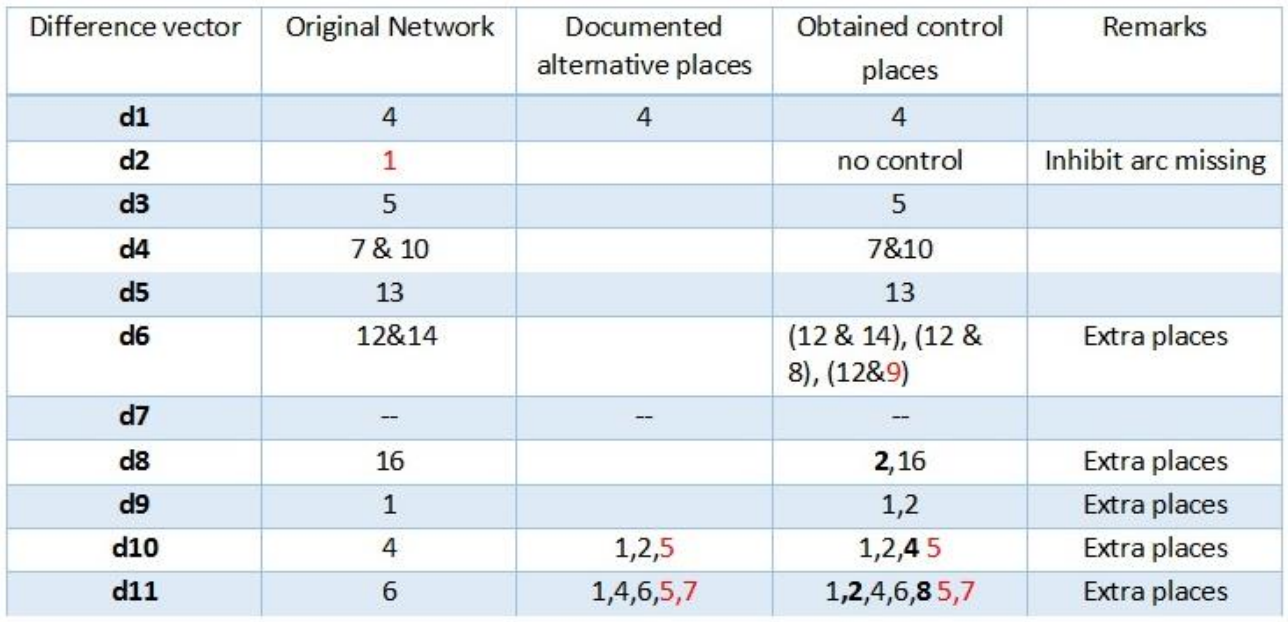
CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 PN RECONSTRUCTION - MATLAB RESULTS

A total of 640 different incidence matrices are conformal with the phosphorylation network's time series data given in [3]. These incidence matrices vary in the input places, output places of transitions besides the total number of transitions in the network. The minimal network in this case has 11 transitions with input and output places which are in complete agreement with the original petri net which was used to model the system.

For the minimal net we find the following alternatives for control places of various transitions



Difference vector	Original Network	Documented alternative places	Obtained control places	Remarks
d1	4	4	4	
d2	1		no control	Inhibit arc missing
d3	5		5	
d4	7 & 10		7&10	
d5	13		13	
d6	12&14		(12 & 14), (12 & 8), (12&9)	Extra places
d7	--	--	--	
d8	16		**2**,16	Extra places
d9	1		1,2	Extra places
d10	4	1,2,5	1,2,**4** 5	Extra places
d11	6	1,4,6,5,7	**1**,**2**,4,6,**8** 5,7	Extra places

Table 4.1 : Results from the MATLAB implementation

The arcs shown in red(BOLD) represent inhibitor arcs.

For the parts of the network which are well specified the network has been reconstructed meticulously i.e. the input and output places match correctly and we exactly know the control places – e.g. for transitions labelled d1, d3, d4, d5, d7. For other parts of the network the control places couldn't be determined precisely – i.e. either there are alternative control places which can explain all the data about the system that we have or the control arc (present in the original network) is not needed to explain the system data. This ambiguity arises due to insufficient time series data of the system. As more observed states and terminal vectors are obtained and added to the system, the ambiguity around underspecified parts of the network diminishes.

This approach is a heuristic free, data driven and analytic approach and has **great potential for design of experiments. After each experiment we know which parts of the networks are ambiguous and next experiment can be designed to better determine these parts!**

4.2 LGTS RECONSTRUCTION - PROLOG RESULTS

The system focuses at finding the minimal LGTS structure that can explain all the system states data observations so far. This includes inferring a FSM which is in some ways a logical equivalent of the k bounded petri net. Besides this, for each control arc of the extended PN, we learn a guard function which is like a constraint box. The constraint box puts a check on the system from making a transition from one state of FSM to other. These constraints are in the form of linear and logical constraints (Booleans). The system has been implemented in the form of an acceptor FSM. It tells you if it is possible to find a LGTS model explaining all the system states thrown at it.

The state vectors were converted to predicate form before feeding them to the system. The following table shows the minimal LGTS networks learnt by the system when fed with the simulation data given in [3].

Difference Vector	Obtained Guards	Original Network Guards
d1	4	4
d2	**--**	**1**
d3	5	5
d4	(7,10)	7 & 10
d5	13	13
d6	One of (12,14) (**11**,8) (**11,9**) (12,8) (12,**9**)	12&14
d7	--	--
d8	One of 2,16	16

d9	One of 1,2	1
d10	One of 1,2,4,**5**	4
d11	One of 1,2,4,5,6,7,8,**9**	6

Table 4.2: Results from LGTS implementation

The guards shown in red(boldface) represent inhibitory interactions. For the guard to evaluate to true these components should be absent. In other words, these appear in the logical/Boolean constraints as $\bar{A}$, while the others appear as true. Since the inference conditions and biological constraint imposed on this system are roughly the same as those imposed on the MATLAB system, as expected the results are in good agreement with the MATLAB code implemented in the first leg of the project.

Qualitative approach gives more freedom of expression but at the same time creates a room for ambiguities and spurious solutions. Incorporation of quantitative information is another direction of innovating besides the incorporation of concurrency and stochastic behavior of the system. Despite this ILP is an efficient way to infer petri net models of a system. This is mainly because search in ILP is more efficient, specially, when it comes to enumerative techniques of searching. Also, it gives us the power to include keep well-established network models in picture. The biological literature that exists can be easily incorporated in the system [25]. This is not possible with most other techniques of inferring models for biological systems.

4.3 COMPLEXITY OF FINDING CONTROL ARCs

The problem in this thesis was a little different from that of minimizing a karnaugh map/Bpoolean function. Here we trying to figure out the minimal solutions that can explain all the observations. If m is the total number of vectors in the state matrix, n is the number of prime implicants, then the algorithm takes **O(n*m)** to find the possible single variable prime implicants , while to figure out the pairs of prime implicants the time taken is of the $\mathbf{O(n^2 * m)}$. However, the latter is found only if no single variable terms could be found. In general to find a term of k variables the time complexity is of the order $\mathbf{O({}^{n}C_{k} * m)}$.

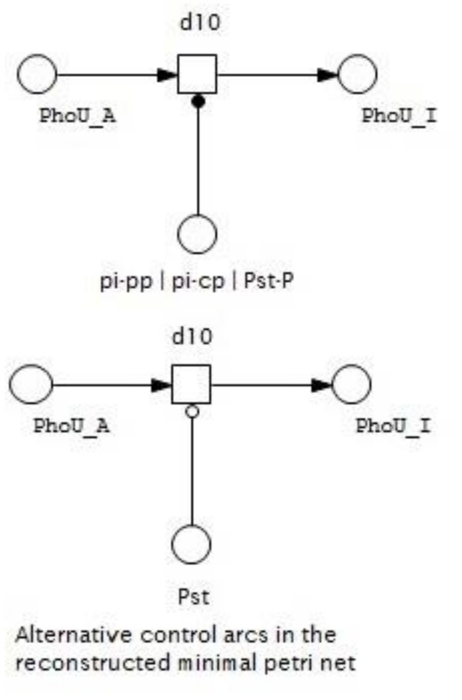

d10
PhoU_A
PhoU_I
pi-pp | pi-cp | Pst-P
d10
PhoU_A
PhoU_I
Pst
Alternative control arcs in the reconstructed minimal petri net

Fig 4.1 (a), 4.1(b)

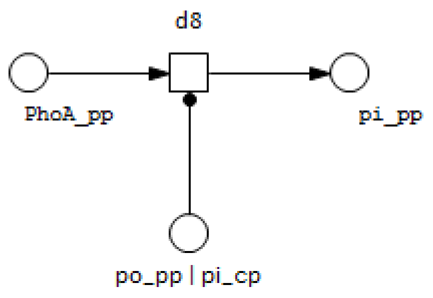


Figure 4.2 : Alternative control arcs in the reconstructed minimal petri net

For $\mathbf{d^2}$, I get that $X_T(d^2)$ = Æ empty set. And hence no need of a control function arises (same as the case for transition d^7 – PhoA triggers production of PhoA-pp). But this is inconsistent with place pi-pp being the catalyst for this transition. However, if **state vector with index = 11 (table 3, last vector of Exp1)** is considered to be a terminal vector, then the set $X_T(d^2)$ contains this vector and the minimal control function evaluates to two possible choices – pi-pp being an inhibitor place (consistent with the original network) or PhoA-pp being inhibitor places.

Again in $\mathbf{d}^{11}$, the extra-minimal control functions ***vanish*** when T=11 is used as a terminal vector for control function evaluation. (The 2 extra minimal control functions being place pi-cp or PhoR acting as catalysts).But this state vector index = 11, last vector of Exp1 has not being explicitly mentioned as a terminal vector. It is just the last vector in the last vector of the first experiment's state matrix where the simulation was abruptly ended and hence should not be taken into consideration as a terminal vector.

For $\mathbf{d}^{6}$, difference vector – 6 (PhoA produced in the presence of catalysts phoA-T and PhoB-P), I get on reconstruction that beside the actual catalysts (PhoB-P and PhoA-T) – there are 2 other minimal control functions

i) PhoB-P and PhoR are catalysts,

ii) PhoB-P is a catalyst while PhoR-P is an inhibitor

Again these are not listed in the journal.

We conclude that the data from the simulation presents some ambiguity about the mechanism of the molecular reactions in the system. More experimental data will help to narrow down on these ambiguous parts of the network and reach a single network.

This chapter sums up the result of network inference and reconstruction from the two implementations carried out. We have also presented and discussed the results from the novel control function inference problem. In the last chapter we present the conclusions from this year long project and discuss the scope for future work.

CHAPTER 5

CONCLUSIONS AND SCOPE OF FUTURE WORK

The chapter summaries the work carried out under Major Project Part 1 and Part 2. It also highlights some of the future work that can be undertaken to continue the study of the system considered in the project.

5.1 CONCLUSIONS

1. The motive of the project is to infer models of a biological system from the time series evolution of the system using combinatorial, analytical, artificial intelligence based techniques in a logical and data-driven fashion. No heuristics or biases have been incorporated to generate these models. Models of biological systems are complex and difficult to infer, however, very crucial for efficacious drug design and delivery.
2. There are 3 main parts in this project. In the first part, after an extensive literature survey we decided to model the system using Petri-nets along the lines proposed in [3]. The system we implemented can infer extended petri net models given the time series evolution data of a biological system. Therefore from the system implemented in MATLAB, it is possible to generate one or more than one extended Petri-net structures that are conformal with the monotone time series data describing a system and hence can explain the system dynamics.
3. In the next leg of the project we developed and implemented an approximate algorithm to find control functions for each transition. This is different from the Quine McCluskey used listed in literature. It has a time complexity $O(k *(m+n))$ where k is the number of components, m the no of states where the reaction is known to fire and n is the number of terminal vectors at which the reaction vector fires.
4. Finally, a PROLOG implementation was carried to infer LGTS models of biological systems. A LGTS is a richer model than a PN for the networks under consideration, however the two have the same turing complexity.
5. The two implementations have been benchmarked by inferring/reconstructing model of the phosphate regulatory system in enteric bacteria. The MATLAB code lists the exhaustive set of conformal PNs while the LGTS system infers the minimal net.

5.2 SCOPE FOR FUTURE WORK

Systems Biology is an emerging area where biological systems are studied in terms of their interactions to explain the functioning as well as the behavior of these systems. Some of the things which are not put into consideration during the commencement of the project and which can later be undertaken as continuation of this work are:

1. A large number of possible petri-net structures are conformal with the data input. However, these arise because of ambiguity in certain parts of the network. The system can be adapted to list out these ambiguous network segments so that the further experiments can be designed to infer those segments particularly.
2. Testing the scalability of these systems is another interesting area.
3. For large systems, with a significant number of interacting species, the number of components listed as catalysts/inhibitors is large. More constraints can be incorporated in the algorithm to make this list smaller and accurate.
4. In this work the number of places in the petri-net is fixed. However there might be short-lived intermediate species that are generated in the reaction mixture and can be taken into account by having a flexible number of places in the petri-net.
5. The LGTS implementation finds the minimal LGTS network that explains the time series data. It can be updated to exhaustively list all possible LGTS networks that conform to the input time series monotone data.
6. 'Automatic Network Reconstruction using ASP' by Durzinsky et al is a variant of the same algorithm but looks more promising and can be easily implemented and tested.
7. The possibility of including prior structural information as constraints for inferring actual gene regulatory networks makes a CLP-based approach particularly appealing.

Appendix

Process	Leads to state
Initial marking, inorganic phosphate present	x0
Uptake of Pi, depleting Pi from the periplasm	x1
Dephosphorylation of PstSCAB-P	x2
Activation of PhoU	x3
Phosphorylation of PhoR	x4
Phosphorylation of PhoB	x5
Biosynthesis of PhoA	x6
Transport of PhoA into the periplasm	x7
Degradation of Po_PP (organic phosphate)	x8
Phosphorylation of PstSCAB	x9
Deactivation of PhoU	x10
Dephosphorylation of PhoB-P	x11

Table: Modelled molecular events that starting from the initial marking, lead to successive states of the Petri-Net [3]

PN vs FSM

Petri nets are inherently non-deterministic and asynchronous and therefore can model concurrency. They can represent the coordination of multiple activities in a system. Unlike FSM they are capable of modelling resource contention, starvation and deadlocks, also race conditions.

FSM model how a system changes in a reaction to internal/external triggers. They are deterministic and model I/O behavior. They cannot model concurrency and hence not every pn can be presented as a FSM. However, PN where each transition has exactly one input and one output can be often modelled as a FSM.

Clearly FSM and a PN are not equally powerful. As a matter of fact every FSM can be modelled by a PN but not vice-versa. However the inter-conversion is possible in some cases. Each (state, input) pair in a FSM converts into a transition in a PN. The FSM being in state x0 and no input corresponds to the initial state of the petri-net.

A PN strictly conservative in preserving the number of tokens can be represented as a FSM! Finite state machines are popularly used to describe the state of a system and the inputs vs outputs of the system. They are not very useful for concurrency or distributed parallel systems. For concurrency, synchronization etc, a more general version of communicating, concurrent state machines should be developed.

BIBLIOGRAPHY

[1] Networks in Nature: Dynamics, Evolution & Modularity, Doctor of Philosophy thesis, Agarwal S, Merton College, University of Oxford, Hilary 2012

[2] Reconstruction of extended petri-nets from time series data & its applications to signal transduction & to gene regulatory networks, Durzinsky et al BMC Systems Biology 2011 , 5:113

[3] An algorithmic framework for network reconstruction, Durzinsky M, Wagler A, Weismantel R, Theoretical Computer Science 412 (2011) 2800–2815

[4] Reconstruction of extended petri-nets using disjunctive control functions, Durzinsky et al Mathematical Biology 2011, 5:113

[5] Learning Petri Net Models of Biological Systems using ILP Ashwin Srinivasan and Michael Bain

[6] Inductive Logic programming: Theory & Methods Stephen Muggleton & Luc De Raedt

[7] Incremental Identification of Qualitative Models of Biological Systems using Inductive Logic Programming , Srinivasan A, R D King , Journal of Machine Learning Research 9 (2008) 1475-1533

[8] A new approach to decoding life: Systems Biology, Ideker et al. 1527-8204/01/0728-0343 14.00

[9] LGTS - Knowledge-guided Identification of Transition-Based Models of Biological Systems using ILP – Srinivasan A, Bain M, and K. Sriram

[10] Websites - en.wikipedia.org/Gene_regulatory_networks

[11] http://en.wikibooks.org/wiki/Embedded_Control_Systems_Design/Finite_State_Machines_and_Petri_Nets

[12] Lecture by Prof P. Devanbu: http://www.cs.ucdavis.edu/~devanbu/teaching/160/docs/petrinet.pdf

[13] Modelling in Systems biology The Petri Net Approach Ina Koch, Wolfgang Reisig, Falk Schreiber (Springer – Computational Biology)

[14] PROLOG – an approach to artificial intelligence by Bratko

[15]Petri net reachability graphs : decidability status of first order properties, darondeau P, Demri S, Meyer R, Morvan C, Logical Methods in Computer Science,Vol. 8(4:9)2012, pp. 1–28

[16] http://en.wikipedia.org/wiki/Petri_net

[17] http://en.wikipedia.org/wiki/Systems_biology

[18] Petri nets in Snoopy: a unifying framework for the graphical display, computational modelling, and simulation of bacterial regulatory networks, Methods in molecular biology (Clifton, N.J.) 01/2012; 804:409-37. DOI: 10.1007/978-1-61779-361-5_21, PubMed

[19] http://en.wikipedia.org/wiki/Gene_regulatory_network

[20] Analysing microarray data in drug discovery using systems biology, Bor-Sen Chen , Cheng-Wei Li, Informa Healthcare, Expert Opinion Drug Discover(2007) 2(5) : 755-768

[21] Gene regulatory network inference: evaluation and application to ovarian cancer allows the prioritization of drug targets, Piyush B Madhamshettiwar, Stefan R Maetschke, Melissa J Davis, Antonio Reverter and Mark A Ragan, PubMed , Genome Med. 2012 May 1;4(5):41. doi: 10.1186/gm340

[22] http://medicine.tamhsc.edu/graduate-studies/faculty/systems.html

[23] http://web.iitd.ac.in/~sumeet/research.html

[24] Unveiling the role of network and systems biology in drug discovery, Albert Pujol, Roberto Mosca, Judith Farrés, Patrick Aloy Volume 31, Issue 3, March 2010, Pages 115–123

[25] Knowledge-Guided Identification of Petri Net Models of Large Biological Systems, Ashwin Srinivasan, Michael Bain, ILP 2011: 317-331

[26] Hsieh Y-J, Wanner BL: Global regulation by the seven-component Pi signaling system. Current Opinion in Microbiology 2010, 13(2):198-203.

[27] Neidhardt FC, Ingraham JL, Schaechter M: Physiology of the Bacterial Cell. A Molecular Approach. Sunderland, Massachusetts: Sinauer Associates;1990

[28] Marwan W, Rohr C, Heiner M: Petri nets in Snoopy: A unifying framework for the graphical?display, computational modelling, simulation, and bioinformatic annotation of bacterial regulatory networks. In Bacterial Molecular Networks. Edited by: Thieffry D. Humana Press;

[29] Carl Adam Petri and Wolfgang Reisig (2008) Petri net. Scholarpedia, 3(4):6477.

[30] Kommunikation mit Automaten. Petri, C.A., Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962, Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65--377, Vol.1, 1966, Pages: Suppl. 1, English translation, http://www.techfak.unibielefeld.de/~mchen/BioPNML/Intro/pnfaq.html

[31] An MTBDD-based Implementation of Forward Reachability for Probabilistic Timed Automata Fuzhi Wang and Marta Kwiatkowska, In *Proc. 3rd International Symposium on Automated Technology for Verification and Analysis (ATVA'05)*, volume 3707 of Lecture Notes in Computer Science, pages 385-399, Springer. October 2005.

[32] Coloured Petri nets, Discrete Event Systems: A New Challenge for Intelligent Control Systems, IEE Colloquium on 4 Jun 1993, 5/1 - 5/3, 4518815, IET

[33] http://daimi.au.dk/CPnets/proxy.php?url=/CPnets/intro/verybrief

[34] Discrete, Continuous, and Hybrid Petri Nets By René David, Hassane Alla

[35] Continuous and hybrid petri nets, Hassane Alla and Rene David, J CIRCUIT SYST COMP 08, 159 (1998). DOI: 10.1142/S0218126698000079

[36] Gene regulatory networks, Eric Davidson, Michael Levin, vol. 102 no. 14 Eric Davidson, 4935, doi: 10.1073/pnas.0502024102

[37] Kauffman, Stuart (1993). *The Origins of Order*. ISBN 0-19-505811-9.

[38] Kauffman SA (1969). "Metabolic stability and epigenesis in randomly constructed genetic nets". *Journal of Theoretical Biology* 22 (3): 437–467. doi:10.1016/0022-5193(69)90015-0, PMID 5803332

[39] Relationship between probabilistic Boolean networks and dynamic Bayesian networks as models of gene regulatory networks, 2006, Lahdesmaki H, Hautaniemi S, Shmulevich I, Yli-Haria O, 814-834, 17415411

[40] Knowledge-Guided identification of petri net models of large biological systems by Ashwin Srinivasan and Michael Bain, ILP'11 Proceedings of the 21st international conference on Inductive Logic Programming, P ages 317-331, Springer-Verlag Berlin, Heidlberg 2012, ISBN: 978-3-642-31950-1,10.1007/978-3-642-31951-8_27

[41] Life, logic and information, Paul Nurse, N*ature* 454, 424-426 (24 July 2008) | doi:10.1038/454424a; Published online 23 July 2008

[42] Using protein-protein interactions for refining gene networks estimated from micro-array data by bayesian networks, Narai et al

[43] http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/

[44] http://en.wikipedia.org/wiki/Espresso_heuristic_logic_minimizer

[45] http://en.wikipedia.org/wiki/Quine%E2%80%93McCluskey_algorithm

[46] Muggleton, S.; De Raedt, L. (1994). "Inductive Logic Programming: Theory and methods". *The Journal of Logic Programming*. 19-20: 629–679. doi:10.1016/0743-1066(94)90035-3

[47] http://en.wikipedia.org/wiki/Prolog

[48] http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html

[49] Prolog Programming for Artificial Intelligence by Ivan Bratko