Modelling Bacterial Regulatory Networks With Petri Nets

Thesis by

Oliver J Shaw

In Partial Fulfillment of the Requirements

for the Degree of

PhD

University of Newcastle upon Tyne

Newcastle upon Tyne

2007

(Submitted June 19, 2007)

To Mum, Dad and Si

Acknowledgements

I would like to thank all those that have helped and supported me through this dissertation. Many thanks go to my two supervisors Dr Jason Steggles and Dr Anil Wipat. Their constant support and enthusiasm made this thesis possible. Members of the bioinformatics lab at Newcastle, past and present, have provided much support and inspiration: Frank Gibson, Tracy Craddock, Keith Flanagan, Peter Li, Richard Banks, Matt Pocock, Philip Lord, Allyson Lister and Jennifer Hallinan. Thanks also go to others who have helped my research with many interesting discussions: Nick Allenby, Victor Khomenko, Maciej Koutny, Albert Koelmans, Darren Wilkinson, Colin Gillespie, Conor Lawless, Daryl Shanley and Carol Proctor. The school of Computing science at Newcastle, exemplified by Professor Pete Lee has given me much support in the pursuit of this dissertation, providing a nurturing yet challenging environment for young researchers. Financial support for this thesis was provided by BBSRC grant 02/B1/X/08298.

In addition to those mentioned above, thanks go to my friends for keeping me sane during this time: Michael Ratter, Martin Ramsdale, Einar Volset, Charles Kubicek, Simon Woodman and Dan Owen. Finally, special thanks go to Taryn Hoole, for all her love and support.

Abstract

To exploit the vast data obtained from high throughput molecular biology, a variety of modelling and analysis techniques must be fully utilised. In this thesis, Petri nets are investigated within the context of computational systems biology, with the specific focus of facilitating the creation and analysis of models of biological pathways.

The analysis of qualitative models of genetic networks using safe Petri net techniques was investigated with particular reference to model checking. To exploit existing model repositories a mapping was presented for the automatic translation of models encoded in the Systems Biology Markup Language (SBML) into the Petri Net framework. The mapping is demonstrated via the conversion and invariant analysis of two published models of the glycolysis pathway.

Dynamic stochastic simulations of biological systems suffer from two problems: computational cost; and lack of kinetic parameters. A new stochastic Petri net simulation tool, *NASTY* was developed which addresses the prohibitive real-time computational costs of simulations by using distributed job scheduling. In order to manage and maximise the usefulness of simulation results a new data standard, *TSML* was presented. The computational power of NASTY provided the basis for the development of a genetic algorithm for the automatic parameterisation of stochastic models. This parameter estimation technique was evaluated on a published model of the general stress response of *E. coli*. An attempt to enhance the parameter estimation process using sensitivity analysis was then investigated.

To explore the scope and limits of applying the Petri net techniques presented, a realistic case study investigated how the Pho and σ^B regulons interact to mitigate phosphate stress in *Bacillus subtilis*. This study made use of a combination of qualitative and quantitative Petri net techniques and was able to confirm an existing experimental hypothesis.

Contents

Chapter 1

Introduction

We are now in a post-genomic age; with advances in high-throughput biological techniques making whole genomic experiments possible [GCN^{+}02]. From shotgun clustering techniques to sequence alignment methods, mathematical and computer science techniques have played an important role in the advancement of biology. It is now possible to sequence a complete genome in a matter of weeks [KO^{+}97]. With the genome available, experimental techniques such as microarrays can simultaneously monitor expression levels of every gene in an organism [SSDB95]. Post genomic techniques such as microarrays are capable of producing vast amounts of data. Indeed, the future investigation of biological systems will greatly dependant on novel theoretical techniques to manage and interpret this data. For example *Bacillus subtilis* [HCW01] contains over 6000 genes [KO^{+}97]. Information on the dynamics of 6000 genes is clearly too large to be analysed and interpreted by the human eye. Hence the rapidly maturing fields of Bioinformatics [WPT02] and Systems biology [Kit01] will play important roles in the future understanding of life. Varied theoretical techniques, both new and well studied, need to be applied to further our understanding of biological systems. The Bioinformatition will fill a key role interfacing between the advanced mathematical, statistical and computing science techniques and the wet lab biologists, with their increasingly sophisticated high throughput techniques. Biological network modelling will form a large component of these integrative studies. A conceptual model of a biological system *in silico* allows a researcher to move from reductionist biology through to understanding of how a whole pathway or even a whole cell functions. A well constructed model in the right formalism will allow a researcher to ask many questions of the model, from the validation of current understanding through to the generation of hypotheses testable in the laboratory. The interdisciplinary work of this nature will no doubt be a key factor in the advancement of biological sciences over the coming years.

As integrative studies begin to gather momentum it is important to have a suitable modelling framework in which to investigate and analyse the resulting biochemical network models. This thesis argues that Petri nets are such a formalism. Petri nets combine a simple graphical notation with powerful, flexible analysis techniques. This thesis begins with the post genomic age and seeks to investigate computational systems biology within the field of Petri nets [Pet62]. Petri nets and accompanying techniques will be investigated with the aim of applying these techniques to advance systems biology. To put the work into context, a brief historical background follows.

1.1 Bioinformatics

As molecular biology techniques advanced appropriate computational methods and resources were required to cope with the ever increasing volume of data. Ever more CPU power and memory was needed to process this data. With regard to computer hardware Moore's law continued to be true in all aspects of computer hardware. It is now possible to buy hard disk space at less than 50 pence a Gigabyte, several thousand times cheaper than 20 years ago. As molecular biology developed, the increasing amount of data meant that problems became more computational. Furthermore, methods for storing data, as well as analysing and comparing it became a challenge. The investigation of molecular biology using computational methods is known as bioinformatics [OV03]. Bioinformatics can be defined as, "The development and use of computational and mathematical methods for the acquisition, archiving, analysis and interpretation of biological information to determine biological functions and mechanisms as well as their applications in user communities" [BBS05]. Bioinformatics has become a field of research in its own right, with many papers and books being published and researchers around the world tackling its problems. In the 1990s Bioinformatics dealt mainly with the management and analysis of DNA, RNA and protein sequences. Now Bioinformatics covers many more areas of research, including but not limited to, physical protein structures, prediction of gene networks and protein protein interactions [WPT02].

1.2 Computational Systems Biology

With the emerging success of Bioinformatics and molecular biology, biological experiments are possible on a genomic scale [GCN$^+$02], via the use of technologies such as microarrays [SSDB95]. Advances in biology alongside the advanced computational techniques and effective data storage provided by bioinformatics has provided researchers with the ability to study multiple components of biological systems. It is now important to appropriately investigate how biological components interact and collectively produce the behaviour of the system as a whole [IWK$^+$04]. Studies seeking to elucidate the behaviour of biological systems often involve a cycle between laboratory experiments, bioinformatics and modelling components. Studies such as these form what is known as "systems biology". Systems biology can be defined as, "...an approach of study where the goal is to understand how the parts of a (biological) system interact to yield a the behaviour of that system as a coherent whole"[IWK$^+$04]. Systems biology is an emerging field, exemplified by the creation of a systems biology dedicated journal [IWK$^+$04].

Systems biology often involves the combination of work by laboratory scientists, computing scientists and statisticians. One aspect of these studies is the creation and analysis of models representing the biological systems of interest. Systems biology studies ideally involving a cycle between model development and analysis allied to biological experimentation. As the model is created, laboratory experiments help fill in the gabs by collection of necessary data. When the model is mature, it can be utilised to make predictions and help direct laboratory experiments. With regard to modelling there are a number of techniques that can be applied to systems biology studies. These techniques include, amongst others: Boolean based modelling [Kau69]; deterministic simulations [IBG$^+$04]; stochastic simulations [MA97]; and network validation [HK04]. While there are many methods that provide great insight to biological systems, they are all targeted at a specific problem domain (for example high level Boolean modelling, or detailed kinetic simulations). Hence the main drawback of the techniques such as these currently applied in systems biology is the lack of flexibility. Ideally a suitable framework for the investigation of computational systems biology will combine intuitive graphical notations, powerful analysis techniques, theoretically justified simulation methods and a mature tool support. In this thesis it is argued that the established Petri net formalism [Pet62] should be adopted as a suitable vehicle for systems biology studies.

1.3 Petri Nets

.

Petri nets [Pet81] are a well studied mathematically based modelling framework originating from computer science. As well as providing an intuitive graphical notation, Petri nets provide powerful analysis and simulation algorithms and techniques (see [PNW04] for a bibliography). The creator of Petri nets, Carl Petri, envisaged right from the start their use for modelling biological Phenomena [Pet62].

Petri nets are bipartite graphs consisting of the following components:

- *Places,* to represent resources, or inputs into a reaction;
- *Transitions* to represent reactions or events;
- *Directed Arcs* to link places to transitions or transitions to places;
- *Tokens* which are distributed over the places, and represent a measure of quantity or concentration.

Petri nets have been used to study a number of different phenomena including hardware design [YK98], industrial manufacturing processes [DHP+93, ZD90], communications networks [BR99], system verification [Kho03], verification of distributed algorithms

[DK98], Local Area Networks [MNCV00] and neural networks [Zar88] to name a few. In recent years they have been used to model biological phenomena [RLM96], and we discuss this area in detail in Chapter 2. Petri nets provide a flexible and powerful framework for the analysis of biological networks. Within the same formalism analysis can be carried out at different levels depending on the information available. Petri nets provide, among others, the following key features to assist the analysis of a network:

- An intuitive graphical representation of networks;
- The network structure can be analysed for structural properties, elucidating such phenomena as feedback loops and providing validation of a model for further analysis;
- Given the initial amounts of molecules or proteins reachability analysis can be carried out to determine the reachable states of the system;
- By including the notion of time, the network can be dynamically simulated to produce time series data similar to lab experiments.

Other slight variations such as Coloured nets [HKV01] extend the formalisms modelling power even further. Petri nets are discussed in much greater detail in Section 2.

1.4 Aims and Structure of the Thesis

This thesis focuses on the modelling aspect of computational systems biology specifically within the Petri net framework. The thesis investigates the creation, analysis and simulation of biological networks using Petri nets [Pet77]. In particular there are a number of more specific aims:

- **Model composition.** With large models having a great deal of complexity, is the composition of smaller models an effective modelling strategy?

- **Stochastic network analysis.** With stochastic Petri nets having the potential to provide a powerful tool for systems biology, what can be done to assist this approach, particularly with regard to missing parameters and simulation costs.

- **Tool support** is to be investigated. While there are a range of Petri net tools available, how are they suited to the study of biological systems? Will further tool development be required?

- **Standards and interchange.** Is there a way of easily importing system biology models into the Petri net framework? What other standards, if any, are needed to support Petri net modelling of biological systems.

- Finally the thesis aims to present a case study of Phosphate stress response in *Bacillus subtilis* incorporating a number of Petri net techniques and drawing the knowledge of the thesis together.

The remainder of this thesis is organised as follows.

Chapter 2: This chapter presents the background to this work, including relevant techniques and related works. Systems biology is introduced, along with a detailed discussion of stochastic simulation techniques, including the Gillespie algorithms [Gil77, Gil76] and the Gibson-Bruck algorithm [GB00]. Petri nets are discussed in detail, with a formal definition and discussion of relevant analysis techniques. Related studies utilising the Petri net framework in systems biology, are then surveyed and discussed.

Chapter 3 This Chapter is concerned with the Safe Petri nets analysis of Boolean representations of genetic regulatory networks and with the import and analysis of Place transition net models of metabolic networks. Boolean representations of genetic networks are first minimised by logic reduction techniques [BMSSV93], and translated into Petri net structures via techniques presented by Steggles *et al.* [SBSW05]. A novel model of the lac operon in *E. coli* is created using this technique. The model is then analysed, and are found to present accurately the high level understanding of the systems behaviours. A mapping is presented allowing the automatic import of systems biology models encoded in SBML [HF$^+$03] to be imported into the Petri net framework, in PNML format [BCvH$^+$03]. This opens a vast resource of models for analysis with Petri nets. This approach is illustrated by the structural analysis of models converted from the SBML repository [SBM04] and the KEGG metabolic database [KGKN02].

Chapter 4: This Chapter presents a new stochastic Petri net simulation tool, NASTY which combines stochastic Petri nets with the Gibson-Bruck algorithm, providing an efficient tool for stochastic simulations within the Petri nets framework. The investigation of the dynamic, time course behaviour of biological systems is an important aspect of systems biology. With stochastic simulations being computationally expensive [EB01], NASTY utilises a distributed engine, allowing jobs to be carried out over a large number of computers. A novel interchange format, namely Time Series Markup Language (TSML) is then presented. TSML allows results from computationally intensive simulations to be stored in an efficient and complete format. This allows the data to be statistically investigated and transferred between groups. The results format is also computationally amenable, facilitating the storage and manipulation of the data.

Chapter 5: This Chapter presents a probabilistic technique which seeks to parameterise stochastic models with the use of genetic algorithms. This technique addresses a current problem with stochastic models, in that there is often a lack of complete kinetic parameter sets [MS03]. An existing model of the general stress response in *E. coli* [SPB01] is taken as a case study for this technique. The model is stripped of all kinetic parameters, with the model's topology and initial concentrations forming the input to the genetic algorithm. The genetic algorithm utilises NASTY as its simulation mechanism to assess the fitness of parameter sets, evolving populations relative to their closeness to the original time-courses. The algorithm is successful in finding qualitatively suitable parameters for all simulation conditions, and quantitatively suitable solutions for the vast majority. Future improvements to the algorithm are also discussed. To further investigate stochastic network dynamics, sensitivity analysis is applied to the *E. coli* model. The aim here is to determine how individual reaction rates contribute to the overall system performance and, if only approximate reaction rates are know, how much insight sensitivity analysis can give. As such, sensitivity analysis is applied to models with complete and approximate parameter sets. Sensitivity analysis of the complete parameter set demonstrates that the model is highly sensitive to a particular parameter. All other parameters had only a small effect on the models dynamics. The approximate model, using random rates, gave very different results, suggesting great care must be taken when utilising sensitivity analysis to assist parameterisation of a stochastic model.

Chapter 6: This Chapter presents a case study, in which a variety of techniques presented previously are utilised to investigate the phosphate stimulon in *B. subtilis*. The purpose of the case study is to further demonstrate the applicability of Petri nets to systems biology. Of particular interest here is the investigation into the interaction of the Pho and σ^B regulons [PH02]. A novel Boolean model was created in a modular fashion from a number of smaller models. Reachability analysis validated that the model captured the high level behaviour of the phosphate stimulon, giving confidence to create a stochastic model. A stochastic Petri net model of the phosphate stimulon was then created. While this model remained reasonably high level in nature, it captured the important behaviour of the stimulon. Sensitivity analysis carried out on the kinetic parameters suggested that the competition of sigma factors for RNA polymerase could indeed provided an explanation for the phenomenon seen in laboratory experiments, adding weight to the suggested explanation by the experimentalists [PH02].

Chapter 7: Finally, this Chapter reflects on the work presented, reviewing the work and where it fits in the current state of systems biology. Future research avenues that have been uncovered by the investigation of this thesis are discussed, along with a general vision for the future systems biology studies, including the role and effectiveness of Petri nets in this field.

Chapter 2

Background

The recent parallel growth in the fields of molecular biology and computing science and the emerging field of Bioinformatics have opened up many areas of research related to the biological sciences. The central theme of this thesis is the study of biological networks, specifically bacterial metabolic and genetic networks, utilising Petri net techniques. In this section the relevant background to studies relating to this thesis are discussed.

The fundamentals of biological network representation are introduced, followed by a range of analysis techniques relevant to this thesis. Simulation techniques, both stochastic and deterministic are discussed with their assumptions and limitations explained. First the Petri net framework is introduced. Relevant mathematically theory and analysis techniques, including behavioural and structural properties, from the Petri net framework are discussed. Stochastic Petri nets, which extend Petri nets with a time element, are introduced and formally defined. Stochastic Petri nets allow the discrete event stochastic simulation of biological networks and are analogous to other stochastic simulation techniques. An review of the current state of the growing application of Petri nets to computational systems biology is presented. Finally the problem of model interchange and storage of is discussed, with the Petri Net Markup Language (PNML) and the Systems Biology Markup Language

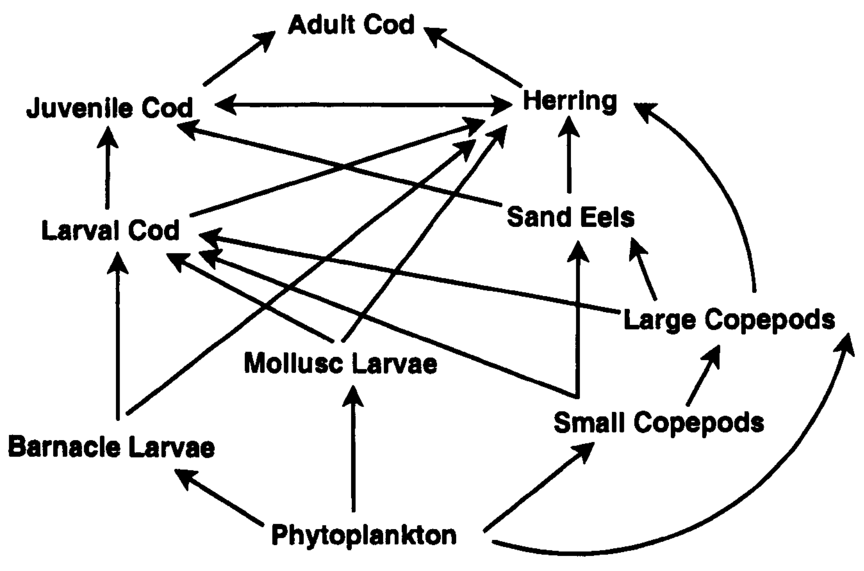


Figure 2.1: A classical marine food web, from [JNC]

(SBML) taking particular prominence.

2.1 Biological Networks

Many phenomena in the biological world can be represented as a network, from classical food webs [JNC], such as that shown in Figure 2.1, to complex protein-protein interactions, as shown in Figure 2.2. These networks are at a basic level made up of nodes, representing a particular state or entity, and arcs, giving the details of node interactions. For example in Figure 2.1, nodes represent organisms involved in the food web, while the arcs represent an organism at the sink of the arc being a consumer of the source of the arc. In a network representation of a biological system the arcs may or may not be directed. For example, in a metabolic network the nodes represent molecules and the arcs would indicate the direction in which the reactions proceed. In protein-protein interaction networks they may

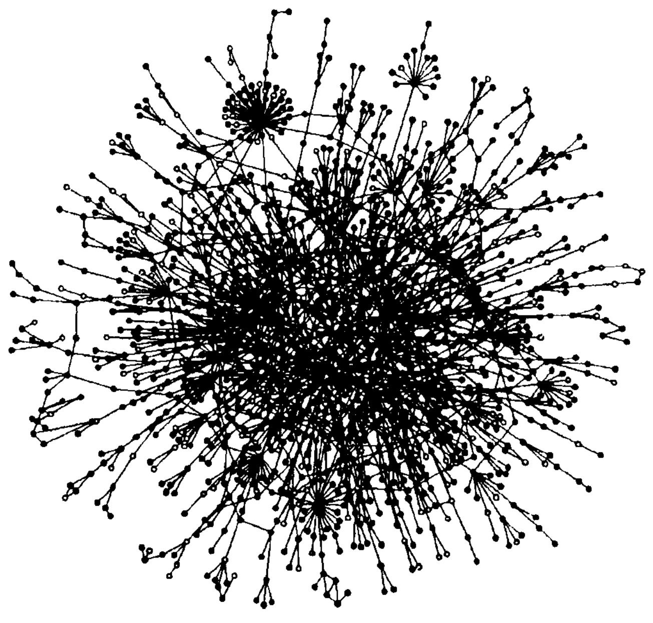


Figure 2.2: Protein-Protein interactions, from [JMBO01]

only represent the knowledge that there is some interaction between one or more proteins, therefore the arc will not be directed. Graphs may also be bipartite, possessing two types of node. In a metabolic reaction for example, a node of type X would represent the reaction while node of type Y would represent reactants and products.

A biological network representation of the molecular state of a cell may consist of many biochemical reactions and interactions. Large networks can be viewed as a number of smaller sub-networks or reactions, with distinct topological characteristics. Each individual reaction is essentially simple but, when combined, the number of reactions and their strong interconnections may manifest the complexity of the networks as a whole [Gib00].

At a most basic level, representing biological systems as networks is a knowledge capture and documentation method. Drawing a model documents understanding and highlights areas where there is uncertainty. If an appropriate modelling framework is utilised modelling biological systems allows the application of many network analysis techniques and algorithms. These techniques can provide great insight into current understanding and also allow the generation hypotheses "in-silico" that may aid the direction of laboratory experiments. Network analysis techniques have a strong history in computing science and are responsible for many important advances. Network analysis techniques are exemplified by the success of Petri nets [Mur89]. Petri nets and many similar techniques, new and old, are increasingly being applied to networks in molecular biology [RML93, PSB+05, GPB+04, CRRT04, AKMM98, IBG+04]. The computational analysis of molecular biological networks is an integral part of the newly developing interdisciplinary field of *systems biology*, which will be discussed next.

2.2 Systems Biology

Traditionally molecular biology has taken a reductionist approach, with experiments investigating the function of a single gene or protein [DDB+04]. *Systems biology* looks to build on this classical work by taking a more holistic view of biology. Systems biology can be defined as, "...an approach of study where the goal is to understand how the parts of a (biological) system interact to yield a the behaviour of that system as a coherent whole"[IWK+04]. The system definition applies at a number of levels: how proteins interact to manifest the behaviour of a cell; how cells interact to manifest the behaviour of a population or a tissue; or how tissues interact to manifest the behaviour of an organism.

Systems biology seeks to utilise expertise from molecular biology, computing science, statistics and bioinformatics. The *in silico* component of these studies are more precisely referred to as *computational systems biology* [Kit02]. Computational systems biology is the creation, analysis and/or simulation of models. Therefore the definition of systems biology must also encompass the creation, definition, storage and simulation/analysis of these models. The aim of computational systems biology is not to capture and model a definitive, complete representation of the system in question. Instead the model is usually an abstract mathematical representation of what are deemed to be the important features and behaviours of the biological system under study [GB01]. The resulting model is then subsequently utilised, via a number of techniques, to interrogate the the system alongside traditional laboratory experiments. Computational modelling within systems biology can reduce the number of experiments needed in the preliminary stages of an investigation, and as such has the following benefits:

- **Cost**. Laboratory experiments use costly equipment and machinery;
- **Time**, Laboratory experiments take a long time to prepare and carry out, while the organism under study may take a long time to grow to a stage required by the experiment;
- **Novel insights**. By attempting to construct a model of the system, features that are currently unknown may become apparent;
- **Repetition**. Many repetitions of experiments on a model can be carried out for very little cost;
- **Documentation**. A full description of the model will be created and stored.

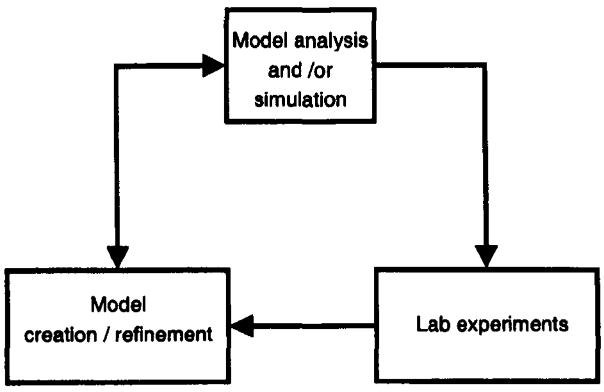

Model analysis and /or simulation
Lab experiments
Model creation / refinement

Figure 2.3: A conceptual view of systems biology, with lab experiments and computational modelling closely coupled

It is imperative that Systems biology be an iterative process. Creation of models must come from real laboratory experiments, molecular biology literature and, where necessary, educated estimates. These models are then analysed and or simulated *in silico.* Results of these *in silico* experiments are compared to laboratory data, adjustments made to the model as necessary, and hypothesis are re-generated from the model (Figure 2.3). For a detailed discussion on the merits and aims of systems biology we refer the interested reader to [Kit02] and [IWK$^+$04]. There are numerous techniques that form a component of computational systems biology. A relevant subsection of these techniques are now discussed.

2.3 Deterministic Simulations

Deterministic simulation techniques are commonly used in systems biology [GHS99]. This is exemplified by the large number of published deterministic models available from the SBML (see Section 2.8) web site [SBM04]. Deterministic simulations utilise Ordinary Differential Equations (ODEs), which are based around tracking the changes in molecular

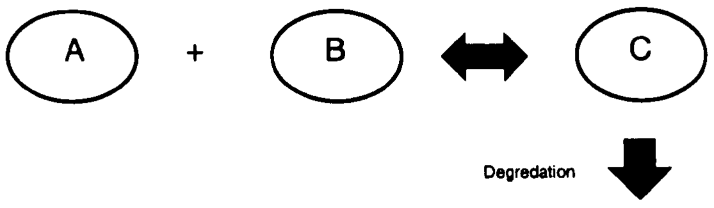


Figure 2.4: A simple system, proteins A and B combine to form protein C, via a reversible reaction. Protein C can then be degraded

concentrations at different time slices. In order to model a system as a set of ODEs the first step is is to write down the reactions involved in the system. The simple system shown in Figure 2.4 is now considered. This hypothetical system involves the combination of proteins A and B to produce the product C. This binding reaction is reversible, allowing protein C to revert back to proteins A and B. Finally protein C can degrade. This simple system can be represented by the following equations:

$$A+B \xrightarrow{k_1} C \tag{2.1}$$

$$C \xrightarrow{k_2} A+B \tag{2.2}$$

$$C \xrightarrow{k_3} \theta \tag{2.3}$$

where A,B,C are biochemical reactants and products, and k_1,k_2,k_3 are kinetic rate constants associated with individual reactions. Kinetic rate constants determine the rate at which reactions proceed. Once the equations are defined, the system can be represented by the *changes* in each reactants concentration at each time slice. Of interest are the concentrations of reactants at a particular time point, where the concentration of any given reactant X is denoted $[X]$. The changes in each reactants concentration at each time slice is represented in the following ODEs:

$$\frac{d[A]}{dt} = k_2[C] - k_1[A][B] \tag{2.4}$$

$$\frac{d[B]}{dt} = k_2[C] - k_1[A][B] \tag{2.5}$$

$$\frac{d[C]}{dt} = k_1[A][B] - k_3[C] - k_2[C] \tag{2.6}$$

By using a number of these equations, and having a knowledge of the initial concentrations and the kinetic rate constants, it is possible to simulate time trajectories of the system. The ODEs can be solved or integrated to find a solution to the system [Dre93]. There are many general techniques for analytically solving ODEs, however these techniques are not practical for the complicated systems presented by systems biology [GB01]. The only viable solution is to use numerical methods, (such as the Runge-Kutta method, the collection method or the Galerkin method), to generate the time course of the reactants in the system [Dre93]. There is a large tool support for modelling biological systems with ODEs and generating their time courses with numerical methods. Tools such as Cellware [DMS$^+$05] and Gepasi [MK98] make ODE modelling techniques relatively accessible to biologists, with little to no theory on the numerical methods needed in order to simulate a system of interest.

The use of deterministic techniques such as ODEs in systems biology rely on a number of assumptions. These assumptions are said to take the "macroscopic" view of the physical world [Gib00]. The main assumptions for deterministic simulation with ODEs are that:

(a) Concentrations vary deterministically over time;

(b) Concentrations vary continuously and continually;

(c) Concentrations are well defined quantities;

(d) Rate constants are well defined quantities.

These assumptions are approximately satisfied if the number of molecules in the system are sufficiently large, hence "macroscopic" chemistry [Gib00]. However these assumptions may not be valid for some important aspects of biological systems. For example, with regard to assumption (a), an analysis of cell protein production (i.e. transcription and translation events) showed that proteins are produced in variable numbers at random time intervals [MA97]. Importantly these variations can lead to large time differences between successive events in regulatory cascades and subsequently produce probabilistic outcomes in switching between alternative regulatory paths [MA97]. These stochastic effects may be a source of some of the unexplained phenotypic variations in isogenic populations [MA97], and deterministic techniques are unable to capture these interesting and important behaviours [SYSY02, MA97]. Assumption (b) breaks down theoretically at the low molecular concentrations found in single-cell based biological systems [Gib00]; while it is reasonable to suggest there is a continuation of concentrations between 6mols/l and 7mols/l, this assumption is not valid under low concentrations as there is no midpoint between 10 and 11 molecules.

Deterministic simulation techniques have proved a successful component of systems biology, particularly in the analysis of metabolic pathways [TPR$^+$00, Kel04]. However, as discussed above, some of the assumptions behind deterministic modelling appear inappropriate for modelling of single-cell genetic regulatory networks [Gib00, ARM98, GB00, SPB01, MA97]. The pioneering work of Gillespie [Gil77, Gil76], has provided alternative, tractable techniques based on more suitable assumptions for mesoscopic systems [Gib00], and has initiated the study of biological networks with stochastic simulation techniques these are discussed in the next section. Finally there are other drawbacks to ODE based modelling. ODE based tool support has improved greatly over the last decade, however the is still a lack of an integrated graphical framework for representing these equations. Other network analysis techniques such as model checking, which may provide crucial information if full kinetic rate constants and initial concentrations are not available, are also absent from the formalism.

2.4 Stochastic Simulations

2.4.1 Assumptions

As previously discussed, deterministic techniques take a macroscopic view of chemical systems. This view is based on the assumptions discussed in the previous section. One alternative set of assumptions is the "mesoscopic" view of biochemistry [Gib00]. The main assumptions behind mesoscopic chemistry can be summarised as follows:

(a) The solution is well-mixed and at thermodynamic, (but not necessarily chemical) equilibrium.

(b) Concentrations change only by discrete numbers of molecules, corresponding to single reaction events.

(c) Single reactions happen at random times.

While these assumptions are very different from those used in deterministic techniques, it should be noted that the stochastic approach is always valid when a deterministic approach is valid and in addition is also valid for certain conditions where the deterministic approach is not [Gil77]. The mesoscopic view of biochemical systems, described in detail by Gillespie [Gil77] and Gibson [Gib00] necessitate different techniques in order to analytically investigate the system of interest. In order to facilitate the future discussions of the stochastic techniques an example system is presented. The same system discussed in Section 2.3 is used to facilitate the discussion, where protein A and protein B can combine to form protein C. Protein C can disassociate back to proteins A and B, protein C is subject to a degradation pathway. As with macroscopic models, mesoscopic models can be represented by a number of equations, each representing a reaction. Note that the reactions are uni directional. If a reaction is reversible it must be represented by two reaction equations. The contrived system described above (Figure 2.4 can be represented by the equations below:

$$A+B \xrightarrow{c_1} C \tag{2.7}$$

$$C \xrightarrow{c_2} A+B \tag{2.8}$$

$$C \xrightarrow{c_3} \theta \tag{2.9}$$

While this is a very simple, contrived system, the techniques are valid for larger systems, with complexity of large networks simply manifests from the additional size of these networks and their strong interconnections [Gib00]. A cursory scan of the equations suggests the difference between these equations and those used to describe the same system in Section 2.3 is the use of c_x instead of k_x as the reaction constants. This is a fundamental and important difference between stochastic and deterministic techniques. Stochastic simulation techniques rely on the assumption that reactions are not determined by constant reaction rates, but as reaction probabilities per unit time [Gil76]. The speed of these reaction is given by the stochastic rate constant c_i. Consider the reaction in Equation 2.7, it can be rigorously argued that c_1 exists, such that $c_1 dt$ is the average probability that a molecular pair will react, according to Equation 2.7 in the next infinitesimal time interval dt. For this it is implied that the probability of some pair of unique molecules of A and B, from the total number of reacting molecules $\#A$ and $\#B$, reacting is given by;

$$c_1 \times \#A \times \#B \times dt = a \tag{2.10}$$

where a is the probability that reaction of type 2.7 will occur in the reacting volume in the next infinitesimal time interval, also know as the "propensity" function.

With the system assumptions defined, the next step is to generate a time-trajectory of components of the model, that is how the behaviour model changes over time [Gil77]. There are two general methods for achieving this: firstly the "master equation" approach [Gil77], and secondly the utilisation of a stochastic simulation algorithm [GB00, Gil77, Gil76]. While the two techniques appear different, "amazingly" their assumptions are provably equivalent [Gib00].

2.4.2 The Chemical Master Equation

As discussed previously, the mesoscopic view of chemistry is founded upon the idea that reactions are governed by the reaction probability per unit time [Gil76], with the state of a system defined by the number of each molecule type in the system, with this state changing whenever one of the reactions occurs [GB00]. The probability of a given reaction occurring moving the system from state S to S' in the next instance of time dt, is given by:

$$P(S',t+dt|S,t) = a_1 dt \tag{2.11}$$

Notice here that the reaction probability depends only on the current state and not on previous states, since time is continuous this is exactly the definition of a Markov process [Mit98] or a Continuous Time Markov Chain (CTMC) [Mar89]. Hence the underlying process can be viewed as a Markov process or CTMC [GB00].

A classical approach for dealing with a system of this type is to construct and solve the "chemical master equation" [Gil77]. Using the argument above, this is the equivalent of solving the underlying Markov process. The key element of the master equation formalism is constructing and solving the "grand probability function" $P(X_1,X_2,...,X_N;t)$ [Gil77], i.e the probability that at time t there will be X_1 molecules of species S_1, X_2 molecules of species S_2...and X_N molecules of species S_N. This can be viewed as a three step process [Gib00]:

- For each possible state of the system S, let $P(S,t|S_0,t_0)$ be the probability that the state is S at time t, given the state is S_0 at time t_0;

- Use the underlying rules of mesoscopic chemistry to specify how $P(S,t|S_0,t_0)$ varies as a function of t;
- The previous steps lead directly to the chemical master equation, a system of linear differential equations with constant coefficients, which when solved gives the time evolution of the reacting system.

The methods for constructing and solving the chemical master equation are beyond the scope of this thesis. The interested reader is referred to [Gib00] for detail of these methods and examples of their use. The chemical master equation approach captures all information in a compact vector equation, while solving it gives the complete probability distribution at any point in time. However, while the master equation is an elegant and powerful approach, it poses insurmountable difficulties for the majority of biochemical systems [Gil77]. The master equation does not lend itself to computational solutions due to the number and nature of the variables involved [Gil77]. For example, a system with three degrading proteins, with initial levels of 100 proteins each, taking part in no reactions other than individual degradations has a state space of 10^6. Solving even this small system requires analysis with a vector of size 1 million and a matrix of 1 million by 1 million [Gib00]. The state space of the model of the lambda phage presented by Arkin [ARM98] leads to a state space of 10^{70}, which is clearly intractable [Gib00]. Worse still, if there is a reaction producing a chemical species, the state space of a Markov chain (and hence the chemical master equation) becomes infinite [Mar89]. For these reasons, stochastic simulation techniques are a necessary alternative [Gil76].

2.4.3 Exact Stochastic Simulation

Stochastic simulation techniques in (bio)chemical systems arose due to the unsuitability of macroscopic chemistry based assumptions for certain, "small" systems and the intractability of the master equation approach [Gil76]. While the master equation approach seeks to solve the probabilities of all possible trajectories, stochastic simulation techniques aim to evaluate a single time time trajectory of the system,(also known as a realisation of the system [GB00]). In order to do this, it is necessary to answer two questions: when will the next reaction occur; and what reaction will that be [Gil77]. Due to the stochastic nature of the system these questions are only answerable in a probabilistic sense [Gil77]. However if these questions are asked using the correct probability distributions, a given realisation of the system will have the exactly the probability that would come out of the solution to the master equation, even if it is not possible to write out the master equation [GB00]. Techniques that generate realisations of the system equivalent to the master equation are know as exact simulation techniques [GB00]. While there are techniques for the approximate simulation of stochastic systems [Gil01], here we concentrate on "exact" techniques.

In order to answer the question relating to when the next reaction will occur, it is necessary that the time between individual reactions called *sojourn* times, is known. The Markovian property requires that the next state to be entered depends only on the current state and not the time spent in it or in any other previous state [Mit98]. This implies that if at any moment the process is observed in state S, the time spent in that state is independent of the time already spent there. This requires a probability distribution function (PDF) with "memoryless" properties, and indeed the Markovian property requires that the sojourn time in states be exponentially distributed random variables [Mar89]. The only PDF with this property is the exponential distribution function [Mit98]. The exponential distribution function is defined as:

$$F(x) = 1 - e^{-\lambda x}; x \geq 0 \tag{2.12}$$

where this function depends on a single variable $\lambda > 0$. The corresponding probability density function is:

$$f(x) = \lambda e^{-\lambda x}; x \geq 0 \tag{2.13}$$

The first and second moments of a random variable X from these functions are [Mit98],

$$E(X) = \int_0^\infty x\lambda e^{-\lambda x} dx = \frac{1}{\lambda} \tag{2.14}$$

$$M_2 = \int_0^\infty x^2\lambda e^{-\lambda x} dx = \frac{2}{\lambda^2} \tag{2.15}$$

hence the residual life time is equal to [Mit98]:

$$E(R) = \frac{M_2}{2E(X)} = \frac{1}{\lambda} \tag{2.16}$$

This states that the remainder of a randomly observed exponentially distributed interval has the same length, on average, as the whole interval [Mit98], also it can be shown that the remainder of the observed interval has the same pdf as the whole interval [Mit98]. All this

Algorithm 1 Gillespie's First Reaction Algorithm

```
1: Initialise initialise rates, molecule numbers, set $t = 0$
2: while Stop conditions are not met do
3:     for all reactions $i$ do
4:         calculate $a_i$
5:         calculate $\tau_i$ using an exponential function with parameter $a_i$
6:     end for
7:     Let $\mu$ be the reaction $i$, where $\mu_i$ is least
8:     Let $\tau \leftarrow \tau_\mu$
9:     Update molecular numbers to reflect the execution of $\mu$, set $t \leftarrow t + \tau$
10: end while
```

implies that the future progress of an exponential distributed activity does not depend on its past. This is know as the "memoryless property", and as previously discussed it is closely related to the Markov property, and is a requirement of the sojourn times of the evolution of a Markov system.

To simulate a realisation of the random variable X the inverse transformational method is utilised [Ros05]:

$$F^{-1}(u) = -\frac{1}{\lambda}\log u \tag{2.17}$$

That is, a random variable u, is obtained with a value between 0 and 1 and use this to obtain exponential sojourns from the propensity functions.

Three predominant exact stochastic simulation techniques are now discussed, namely:

- Gillespie's first reaction method [Gil76];
- Gillespie's direct method [Gil77];
- The Gibson-Bruck algorithm [GB00];

In order to facilitate the discussion of these algorithms, some definitions are required.

The propensity function of a reaction, the probability that the reaction i will occur in the next infinitesimal time step, is defined to be a_i. The putative time until the occurrence of reaction i is defined as τ_i. Finally the reaction that will occur next is labelled μ. A crucial step in any stochastic simulation algorithm is the transition from the value of the propensity function to the putative time until the reaction occurs, as shown in the arguments above, this has to be done utilising the exponential probability distribution function.

The pioneering work of Gillespie is the first algorithm to be discussed. Gillespie's First reaction method (Algorithm 1) [Gil76] calculates the a_i for each reaction i and from this generates a putative time τ_i, until the reaction will occur. To be precise, this is the interval of time before the reaction occurs assuming that no other reactions occur in that time that would alter the system. The reaction with the shortest τ becomes μ, and the system state and time are updated accordingly. This algorithm requires a random number for every reaction $i \in I$, and takes a time proportional to I to update the a_i for all I's and a time proportional to I to find the μ.

Gillespie's Direct method (Algorithm 2) [Gil77] asks which reaction will occur next and when it will occur. This method is very different in approach to the first reaction method, but it is provably equivalent [Gil76]. The direct method is based on the idea that for a given system with m reactions the propensity function for reaction i is a_i, and the propensity of a reaction of any type is:

$$a_0 \equiv \sum_{i=1}^{m} a_i \tag{2.18}$$

From this the time till the next reaction (of any type) can be simulated by sampling

Algorithm 2 Gillespie's Direct Method

```
1: Initialise initialise rates, molecule numbers, set $t = 0$
2: while Stop conditions are not met do
3:     for all reactions $i$ do
4:         calculate $a_i$
5:     end for
6:     Calculate $a_0 = \sum_{i=1}^{m} a_i$
7:     Simulate $t'$ by sampling from the exponentially distributed function of $a_0$
8:     Select a random number, $r$ from the unit interval uniform distribution
9:     select $\mu$ to be the reaction index for which $\sum_{v=1}^{\mu-1} a_v < r * a_0 \leq \sum_{v=1}^{\mu} a_v$
10:    $t \leftarrow t + t'$
11: end while
```

from an exponential distribution of $\sum_{i=1}^{m} a_i$. The gives the time until the next reaction. The reaction type must now be determined. The probability the reaction will be of type i is a_i/a_0. Hence the type of the next reaction can be probabilistically picked. The algorithm is listed in Algorithm 2. This method uses two random numbers per iteration, takes time proportional to the number of reactions to update the a_i's and to calculate a_0.

Gibson's next reaction [GB00] method was devised as an improvement on Gillespie's first reaction method. It "does away" with :

a) updating $a_i \forall I$;

b) generating a putative time $\tau_i \forall I$;

c) identifying τ_μ.

The Gibson-Bruck algorithm (Algorithm 3) achieves these improvements through the intelligent use of a "dependability graph" and an "indexed priority queue". This allows the algorithm to achieve the following:

- **Only recalculate a_i (and hence τ_i) if it changes.** This is made possible by the use of

Algorithm 3 The Gibson-Bruck Algorithm

```
1: Initialise, set initial molecular numbers, kinetic rates and $t \leftarrow 0$
2: for all reactions $i$ do
3:     calculate $a_i$
4:     calculate $\tau_i$ according to an exponential distribution with parameter $a_i$
5: end for
6: while Stopping conditions not met do
7:     store the $\tau_i$'s in an indexed priority queue $P$
8:     set $\mu$ to the reaction whose $\mu_u$ is at the top of $P$ i.e whose putative time is least.
9:     update molecular amounts to reflect the execution of $\mu$, set $t \leftarrow \tau$
10: end while
```

a dependability graph. Hence the reactions that alter a_i is known. If a_i is not changed there is no need to recalculate it.

- **Reuse τ_i where appropriate.** If a_i is not changed then it is not necessary to update τ_i. Conversely if a a_i has changed the memoryless property of the exponential distribution allows us to simply re-sample τ_i from the newly calculated a_i.

- **Switch from relative time to absolute time.** By using an indexed priority queue the differences in times can be stored, hence once the τ_i is in the queue, there is no further book-keeping in updating times when other reactions occur.

Again the Gibson-Bruck algorithm, is provably equivalent to the First reaction algorithm and Gillespie's direct method [GB00]. Hence it is exact and provably equivalent to the master equation approach.

The important algorithms by Gibson-Bruck and Gillespie were developed from withing the field of (bio)chemical reaction systems. Interestingly during this time, and quite independently, stochastic Petri nets [Mol82] were developed within the field of computing science. Stochastic Petri nets also provide an exact method of simulating realisations

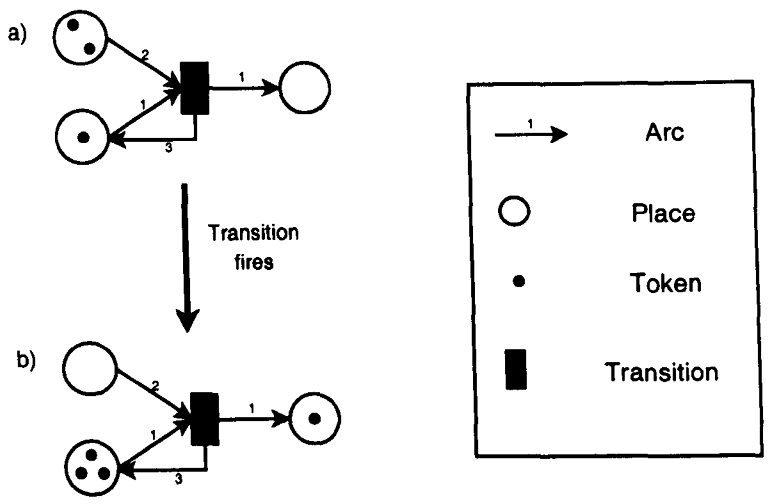


Figure 2.5: Illustration of basic Petri net, including the firing of a transition

of a stochastic system, with an underlying Markov process and hence can be utilised in the investigation of biochemical networks [GP98]. In order to discuss stochastic Petri nets [Mol82], discussed in Section 2.6 Petri nets [Pet62] must be introduced.

2.5 Petri Nets

Petri nets [Pet77] are a formal, mathematical framework for modelling of complex concurrent systems. Petri nets are bipartite graphs made up of *places, transition, arcs* and *tokens* (Figure 2.5). A model of a system represented as a Petri net "...can then be used to evaluate the modelled system and suggest improvements or changes" [Pet81]. Petri nets were originally formalised by Dr Petri in his thesis in 1962 [Pet62]. Since then Petri nets have been extensively studied and, at the most recent count, there were over 8500 papers in the Petri net bibliography [PNW04].

Petri nets have their foundation in theoretical computing science with numerous papers on their fundamental concepts, theory and algorithms. For a listing of these topic and their references see the Petri net Bibliography [PNW04]. Petri net modelling has been applied in a wide range of disciplines. including manufacturing [DHP+93], hardware design [YK98], business process management [LO03], verification of distributed algorithms [DK98], local area networks [MNCV00], neural networks [Zar88] and more recently biological networks [HKV01, GP98, SKSW04a, RLM96]. Petri nets have a number of desirable features as a modelling framework:

- A clear graphical visualisation of the system;
- The ability to analyse the network topology and to determine structural properties;
- The ability to analyse the network topology and state to determine behavioural properties;
- The ability to simulate a network by the addition of a time element;
- The ability to analyse a model with high level variants of a Petri net.

Basic theory for Petri nets is now introduced. For a more detailed introduction the reader is referred to [Mur89].

2.5.1 Place Transition Nets

The most common form of a Petri net is a Place Transition net (P/T net). A P/T net is a bipartite graph consisting of *places, arcs, transitions* and *tokens* (Figure 2.5). A P/T net has places and transitions connected by directed arcs. An arc can connect a place to a transition, or a transition to a place, but is not permitted to connect a place to a place, or a transition to a transition. An arc is given a weight which signifies the replication of that arc (Figure 2.6). The connectivity of the places and transitions gives us the structure, or *topology* of the network. Each place in a P/T net has a non negative number of tokens. Tokens signify the number of the particular resource represented by the place. The number of tokens in a particular place is called the place's *marking*. The collection of all place markings is referred to as the *marking* of the Petri net and this represents the current state of a Petri net. A transition is said to be *enabled* in a given marking if all of its input places have a marking greater than the weight of the arc. Enabled transitions can *fire*, this represents an event, or reaction happening, this alters the marking of the net. Firing a transition removes a number of tokens from the input places (specified by arc weight) of that transition, and adds a number of tokens to the output places (again specified by arc weight).

A place transition net, its transitions, inputs and outputs can be interpreted in a number of different ways. Usually places represent conditions or resources, while transitions generally represent events or processes. A number of different examples are shown in Table 2.1. This variety of interpretations displays how flexible the formalism is. Many different models can be constructed and analysed with the same rigorous mathematical techniques and algorithms. A formal definition is now given for P/T Petri net:

Input Places	Transitions	Output Places
Preconditions	Events	Post-conditions
Input Data	Computational Step	Output data
Input signals	Signal processor	Output signals
Resources needed	Task or Job	Resources released
Conditions	Clause in logic	Conclusions
Buffers	Processor	Buffers
Reactants	Biochemical Reaction	Products

Table 2.1: Some interpretations of Transitions and Places (Modified from [Mur89])

A Petri net *PN* can be described more formally as a 5-tuple $PN = (P, T, F, W, M_0)$ where:

$P = \{p_1, p_2 \ldots., p_m\}$ is a finite set of places,

$T = \{t_1, t_2 \ldots., t_n\}$ is a finite set of transitions, such that $P \cap T = \oslash$ and $P \cup T \neq \oslash$,

$F \subseteq (P \times T) \cup (T \times P)$ is the flow relation (arcs), where $(x, y) \in F$ denotes an arc from x to y,

$W : F \rightarrow \{1, 2, 3 \ldots.\}$ is the weight function (default: 1),

$M_0 : P \rightarrow \{0, 1, 2, \ldots.\}$ is the initial marking of the Petri net.

For any transition t we say p is an *input place* for t if $(p, t) \in F$. Conversely, p is called an *output place* if $(t, p) \in F$. The set of all input places to a transition is denoted $\bullet t$ and is called the *preset* of transition t. More formally defined as:

$$\bullet t = \{p \mid (p, t) \in F\}$$

Similarly the output places of a transition t are called the *post-set* of t, denoted $t\bullet$, and is formally defined by:

$$t\bullet = \{p \mid (t, p) \in F\}$$

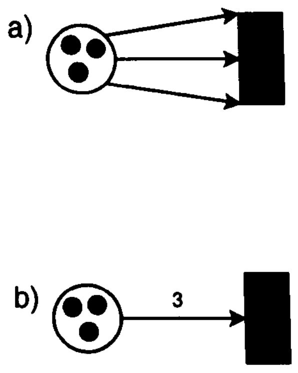


Figure 2.6: Arc weights represent replication of arcs. Hence Petri nets a) and b) are equivalent.

A transition t is said to be enabled in a given marking M when every place in $\bullet t$ has a marking greater that the weight of the arc from each place to the transition. That is for every input place $p \in \bullet t$ we have $M(p) \geq W(p,t)$. A transition is then *enabled* to fire. However there is no guarantee when or if an enabled transition fires. When a transition fires the nets marking changes as tokens are removed from input places and added to output places. The dynamic behaviour of the system is modelled by the change of state or marking of the net. A transition $t \in T$ in a marking M may fire according to the following rules:

1. A transition is enabled (able to fire) if each input place contains at least as many tokens as the input arcs weight i.e every place $p \in \bullet t$ satisfies $M(p) \geq W(p,t)$;

2. Firing a transition removes $W(p,t)$ tokens from each input place, $p \in \bullet t$, of t and adds $W(t,p)$ tokens to each output place, $p \in t\bullet$ of t.

Given a marking M and an enabled transition t in M we can formally define the marking resulting from firing t can formally be defined as follows:

$$M'(p) = \begin{cases} M(p) \text{ if } p \notin \bullet t \text{ and } p \notin t\bullet \\ M(p) - W(p,t) \text{ if } p \in \bullet t \text{ and } p \notin t\bullet \\ M(p) + W(t,p) \text{ if } p \notin \bullet t \text{ and } p \in t\bullet \\ M(p) - W(p,t) + W(t,s) \text{ if } p \in \bullet t \text{ and } p \in t\bullet \end{cases}$$

The firing of a transition t leading from marking M to marking M' is denoted as $M \xrightarrow{t} M'$. A sequence of firings is called an *occurrence* or *firing* sequence. A firing sequence may be finite or infinite. A firing sequence $\delta = t_1, t_2, ...t_n$leads from $M_0$to M_n if:

$$M_0 \xrightarrow{t_1} M_1 \xrightarrow{t_2} M_2 ... \xrightarrow{t_n} M_n$$

This defines a standard Place/Transition net. With the net formally defined it is possible to analyse the net to deduce interesting properties and these are discussed in the following sections.

2.5.2 Structural Properties

Having formally defined a Petri net formal analysis techniques are possible. A net can be analysed simply from a knowledge of the graph structure, that is the network topology. Analysis using just this structural knowledge is the analysis of structural properties. Structural properties do depend on the initial marking M_0 only in that the resulting properties hold for any initial marking. The analysis of these structural properties depends greatly on the matrix based techniques, and this is now discussed.

A Petri net ,*PN* with n transitions and m places, can be represented by its *incidence matrix*. The incidence matrix, is an $n \times m$ matrix defined as: $A = \left[a_{ij}\right]$, where a typical entry in the matrix is given by $a_{ij} = w(i,j) - w(j,i)$, that is the weight of the arcs from transitions to places, take away the weights of the arcs to places to transitions. A number of structural properties can be gleaned based upon this incidence matrix, two of the most common are P-invariants and T-invariants.

2.5.3 Invariant Analysis

Both P (Place) invariants and T (Transition) invariants are calculated from the incidence matrix A. A P-invariant x, is calculated as solution to $Ax = 0$, while a T-invariant y, is calculated as $A^t y$, where A^t is the transpose of the incidence matrix A. T invariants are a set of transitions in a given net, that once fired, return the net to its original marking. In terms of biochemical networks, T invariants may represent reversible reactions, or loops in a reaction. More recently Heiner *et al.* [HK04] have used T invariants in order to validate the structure of a metabolic pathway prior to further investigation with more advanced techniques. P invariants represent a set of places where the total number of tokens on those places does not change regardless of which transitions are fired. P-invariants apply to any initial marking and may represent simple biochemical coupling, such as ATP and ADP. P invariants may also represent the conservation of a particular molecular group, such as phosphate, through a chain of reactions [RLM96].

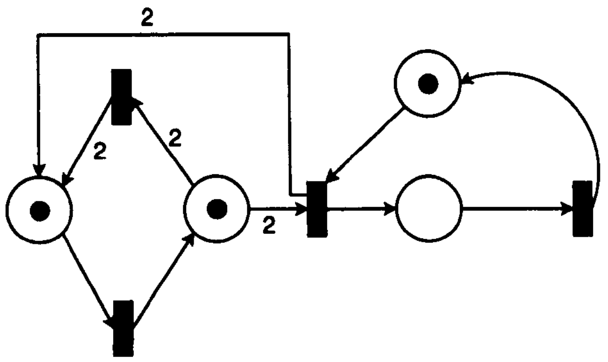


Figure 2.7: A 2-Bounded Petri net

2.5.4 Behavioural Properties

If the topology of the network is known then, as we have seen above many structural properties can be analysed. If, alongside the topology, the initial state of the net is know then *behavioural* properties can be analysed. Many of the behavioural properties depend on the analysis of the nets reachability and coverability trees. Construction and analysis of these trees is discussed in Section 2.5.5.1. Some of the more commonly analysed behavioural properties are now presented.

2.5.4.1 Reachability

Reachability is a key property when studying the dynamics of a system. It determines if it is possible, however likely or unlikely, to reach a given marking from an initial marking. Building on the previously described firing rules we can define reachability more formally. The firing of a enabled transition will alter the marking of a net. A sequence of such firings δ will result in a sequence of markings. A marking M_n is said to be reachable from from M_0 if there exists a firing sequence δ that transforms M to M_n. If this is the case we can

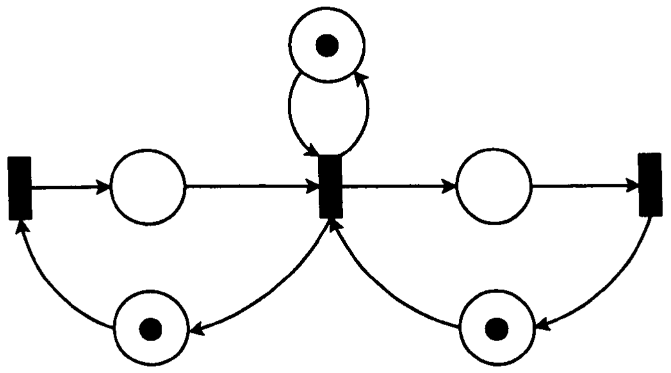


Figure 2.8: A Safe Petri net

write $M[\delta > M_n$. The set of all possible markings reachable from M in a net N is denoted by $R(N,M)$, or $R(M)$. The reachability problem for a marking M_n is to find if $M_n \in R(M)$. Often we may only be interested in the marking of a subset of places, denoted M_n'. This is then known as the sub-marking reachability problem [Mur89], which is the problem of finding if $M' \in R(M)$. The reachability problem has been proven to be decidable [Hac75], although it takes exponential time and space to verify [Lip76].

2.5.4.2 Boundedness and Safe Nets

Boundedness tests if there is an upper limit, k on the number of tokens in a place. If this upper bound applies to all places in the net then the net is said to be k-bounded. More formally we state that a net (N, M_0) is k-bounded if the number of tokens in every place does not exceed k for any reachable marking from the initial marking M_0. More mathematically $M(p) \leq k$ for every place p and every marking $M \in R(M_0)$. Figure 2.7 shows an example of a 2-bounded Petri net in which no place ever has a marking of more than two tokens.

Safeness is a special case of boundedness. A safe net is one where the bound on every place is 1. That is $M(p) \leq 1$ for every p in every marking $M \in R(M_0)$. Safeness is a

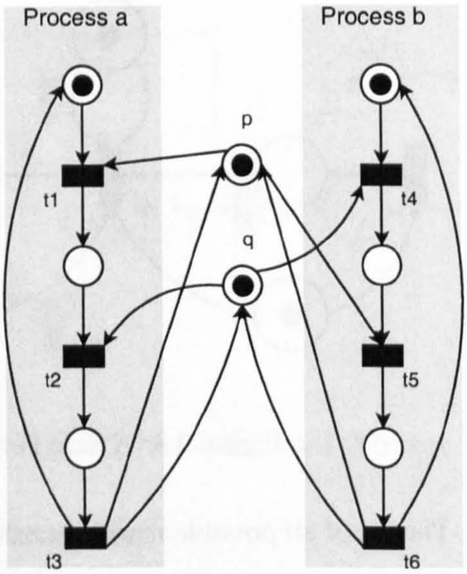


Figure 2.9: Deadlocks in Petri nets

property traditionally used in hardware design due to the obvious benefits of dealing with binary states. Hence safe Petri nets offer an alternative to Boolean networks. The net in Figure 2.8 is an example of a safe Petri net.

2.5.4.3 Deadlocks and Liveness

The study of liveness arose from the problem of dealing with deadlocks [Heb70]. Deadlocks are a common problem in the field of computing science. A deadlock arises when the system reaches a state where no next state is possible, they are a long studied problem, and they typically occur when there is a problem with resource allocation. The Petri net in Figure 2.9 gives an example of deadlock. In this example process *a* and process *b* share the resources *p* and *q*. There are firing sequences that allow both processes to proceed normally, mainly $t_1, t_2, t_3, t_4, t_5, t_6$ and $t_4, t_5, t_6, t_1, t_2, t_3$. However deadlock occurs when the sequence t_1, t_4 is fired, preventing either process from proceeding.

When analysing Petri nets the *absence* of deadlocks, or liveness properties are of interest. A net (N, M_0) is said to be live if in any marking M_n that satisfies $M_0[\delta M_n$ it is possible to fire any of the nets transitions at some future time via another firing sequence δ. Hence a live Petri net guarantees deadlock free operation. As liveness is a costly property to verify, it is often relaxed, and different levels of liveness analysed ([Mur89] and references therein). A transition t in a Petri net (N, M_0) is denoted as:

1. $L0$-live, or dead. If it can never be fired in any firing sequence in $L(M_0)$ where $L(M_0)$ is the set of all possible firing sequences from M_0.

2. $L1$-Live, or potentially fireable. If t can be fired at least once in some firing sequence in $L(M_0)$.

3. $L3$-Live. If given a positive integer k, t can be fired at least k times in some sequence in $L(M_0)$.

4. $L4$-Live, or live. If t is $L1$-live for every marking M in $R(M_0)$.

Coverability is closely related to liveness. A marking M is said to be coverable if there exists a marking M' in the reachability set where the marking of every place in M' is greater than or equal to every place in M. More formally a marking M is said to be coverable if there exists a marking $M' \in R(M_0)$ such that $M'(p) \geq M(p)$ for every p in the net. Finally a net (N, M_0) is reversible if from each marking M in $R(M_0)$, $M_0$is reachable from M.

2.5.5 Petri Net Analysis Techniques

2.5.5.1 Place Transition Net Reachability Trees

Given a Petri net, a *reachability* tree [Pet81] can be constructed. From an initial marking M_0 a new marking for each enabled transition can be obtained. This process leads to a tree representation of the original net. In this tree the nodes represent net markings and the arcs represent the transition fired that move from one state to another. If the net is unbounded then the reachability tree created would be infinitely large. Thus a *coverability* tree [Mur89] can be constructed. The special symbol ω is used. ω can be thought of as infinity. Another problem arises if the net is a loop. In this case the reachability tree would continue to grow infinitely large. To prevent this only new markings (ones not already present as a node in the tree) are considered. Transitions from an old marking are also not considered. Using these rules we can construct a reachability tree from a net using the following algorithm (Algorithm 4) [Pet81, Mur89].

To show how the reachability tree algorithm is used the coverability tree for the Petri net in Figure 2.10 is constructed. The full reachability tree is shown in Figure 2.11. The initial marking given in the net forms the root, (1,0,1,0). From this marking only transition t_3 can fire, this leads to the new state (1,0,0,1). From this state the only enabled transition is t_2. Firing this leads to the state (1,1,1,0), however there is a marking from the root to this state where every place has at least as many tokens, and the two markings are not the same (Algorithm 4 part ii)). Thus the token in p_2 is replaced with a ω, giving us the state $(1,\omega,1,0)$. From here both t_1 and t_3 are enabled. Firing t_1 gives $(1,\omega,0,0)$ which is a dead-end as no transitions are enabled. If t_3 had fired the marking becomes $(1,\omega,0,1)$. Now t_2

Algorithm 4 The construction of a Petri nets Reachability tree

```
1. Label the initial marking $M_0$ as the root, and tag it new
2. While new markings exist, do;
   (a) Select a new marking $M$
   (b) If $M$is is identical to a marking on the path from the root to $M$, then tag $M$ old
       and go to another marking
   (c) If no transitions are enabled at $M$ tag $M$ dead-end
   (d) While there exists enabled transitions at $M$, do the following for each enabled
       transition $t$ at $M$
       i. Obtain the marking $M'$ obtained by firing $t$
       ii. On the path from the root to $M$ there exists a marking $M''$, such that
           $M'(p) \geq M''(p)$ for each place $p$ and $M' \neq M''$, replace $M'(p)$ by $\omega$ for
           each $p$ such that $M'(p) > M''(p)$.
       iii. Introduce $M'$ as a node, draw an arc with label $t$ from $M$ to $M'$, and tag $M'$
            new.
```

is enabled. Firing t_2 gives the marking $(1,\omega,1,0)$, which has already appeared, hence the tree is complete.

2.5.5.2 Structural Reduction

Petri nets can be reduced to a simpler, smaller form to facilitate analysis, and reduce the state space. When a Petri net is reduced it is important that the reduction does not affect the behaviour of the system to be analysed. More common structural reduction techniques are shown in Figure2.12.

The reductions in Figure 2.12 preserve liveness, safeness and boundedness. The properties of liveness, safeness and boundedness are discussed in section 2.5.4. The six reductions in Figure 2.12 are as follows:

- Fusion of Series Places Fig 2.12(a);

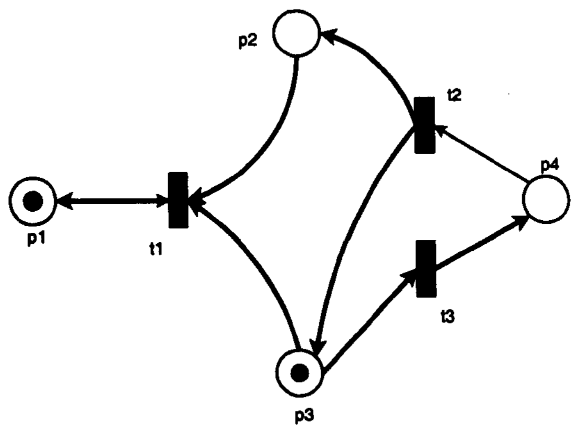


Figure 2.10: A Place Transition net to illustrate the construction of reachability trees

$$\begin{tikzcd}
& (1,0,1,0) \arrow[d, "t_3"] & \\
& (1,0,0,1) \arrow[d, "t_2"] & \\
& (1,\omega,1,0) \arrow[dl, "t_1"'] \arrow[dr, "t_3"] & \\
(1,\omega,0,0) & & (1,\omega,0,1) \arrow[d, "t_2"] \\
& & (1,\omega,1,0)
\end{tikzcd}$$

Figure 2.11: The Reachability tree of Figure 2.10

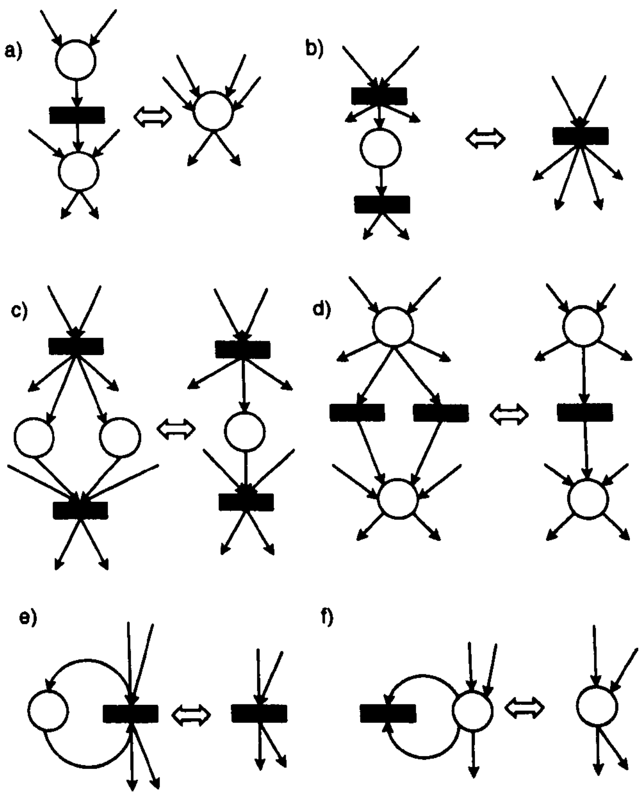


Figure 2.12: Some standard structural reduction rules of Petri nets, from Murata [Mur89]

- Fusion of Series Transitions, Fig 2.12(b);
- Fusion of Parallel Places, Fig2.12(c);
- Fusion of Parallel Transition, Fig2.12(d);
- Elimination of Self-Loop Places, Fig2.12(e);
- Elimination of Self-Loop Transitions, Fig2.12(f).

While these properties are helpful for preservation of boundedness, liveness and safeness, other properties of behaviours may be lost. Hence while these structural reductions reduce the complexity problem for certain properties, care must be taken in ensuring the desired properties hold after a reduction.

2.6 Stochastic Petri Nets

As discussed in Section 2.5, Petri nets provide a powerful, integrated formalism for the analysis of a networks properties in a variety of problem domains. In Section 2.5 the techniques answer questions of the type "Is it possible x can happen" where x can be topics of reachability or the possibility of a reaction happening. These techniques address the "qualitative" issues in model analysis. It is often desirable to ask questions such as, "what happens to the concentration of protein x over 10 minutes" and "what is the system state at time x". These questions are "quantitative" in nature and can only be answered if a notion of time is included as an integral part of the modelling formalism. The issue of adding a timing element to increase the modelling power and versatility led to the creation of

Stochastic Petri Nets (SPNs) [Mol82]. In this formalism Petri nets are extended by the association of an exponentially distributed firing delay with each transition. There are several alternative methods of introducing a timing element into a Petri net, for example time can be associated with either the places or the transitions, the firing semantics can be altered, (e.g atomic or phase firing) and finally the temporal specification may be deterministic or probabilistic. These alternatives will not be discussed further, due to the prominence of SPNs and their inherent suitability to the modelling of biological networks, exemplified by the fact a SPN can be viewed as a continuous time Markov chain [Mar89]. Systems that can be represented by Markov chains have theoretical justification to their application to biological systems [Gil77]. The interested reader is however referred to [Mar89] for other methods of associating time to Petri nets and the various firing semantics therein.

Stochastic Petri nets have been utilised in the analysis of various computing science problems, such as local area network configurations [MNCV00], manufacturing systems [SAS99], work-flows [LQRM02] and analysis of program language execution [RSJ00]. Stochastic Petri nets are models of a probabilistic nature operating in continuous time [Mar89], and hence are analogous to the stochastic simulation techniques discussed in Section 2.2. As such there have been a number of papers on the utilisation of stochastic Petri nets in systems biology. Goss and Peccoud used SPNs to analyse the stabilising effect of the ROM protein on the replication of the Col-El plasmid [GP98]. Srivastava *et al.* utilised similar techniques to investigate the response of the sigma factor, σ^{32} in *E. coli* to heat and ethanol shock [SPB01]. A slight modification of SPNs, stochastic activity networks, were used to study blood coagulation [ML97] and the human menopause [TL02], these studies are discussed further in Section 2.7.

A stochastic Petri net *SPN* can be described more formally as a 6-tuple $SPN = (P,T,F,W,M_0,R)$ where:

$P = \{p_1, p_2 ..., p_m\}$ is a finite set of places,

$T = \{t_1, t_2 ..., t_n\}$ is a finite set of transitions, such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$

$F \subseteq (P \times T) \cup (T \times P)$ is the flow relation (arcs), where $(x,y) \in F$ denotes an arc from x to y,

$W : F \rightarrow \{1,2,3....\}$ is the weight function (default: 1),

$M_0 : P \rightarrow \{0,1,2,....\}$ is the initial marking of the Petri net.

$R = (\lambda_1, \lambda_2, ...\lambda_n,)$ is a set of firing rates (possibly marking dependent) associated with the transitions.

It is noted that the definition of a stochastic Petri net is the same as a P/T net, with the addition of the set of firing rates associated with each transition. In the implementation of a SPN a firing delay is associated with each transition, this delay is a random variable, sampled from the negative exponential pdf, with the parameter being the transitions rate λ.

With the net defined we have two options, either to solve the underlying Markov chain, or to simulate the system utilising an algorithm that produces a random walk with the same probability as the Markov chain [Mar89]. When applied to biological networks, the number of molecules generally implies that the solving the Markov chain is intractable[Gib00]. Hence simulation is the only tractable means of investigating the time evolution of a SPN.

Utilising the memoryless property of the negative exponential (as described in section 2.4) an efficient simulation algorithm can be developed. This is discussed in more detail in section 4 where stochastic Petri nets are amalgamated with the Gibson-Bruck algorithm [GB00] to produces a new tool for the simulation of biological networks.

2.7 Petri Nets and Biology

Biological molecular systems may be viewed as complex concurrent processes. Cellular mechanisms are complex and dynamic, multiple genes are transcribed, multiple types and copies of transcripts are translated to proteins, and those proteins partake in multiple signalling processes and metabolic reactions in a concurrent fashion. Petri net modelling techniques have evolved to model complex concurrent computing systems in a discrete fashion and hence are inherently suitable for modelling biological networks in systems biology applications. For example, Figure 3.6 shows a representation of the glycolysis pathway in *Saccharomyces cerevisiae* as in this Figure 3.6 we see examples of concurrency and choice, features of complex systems from computing science commonly modelled with Petri nets.

Beyond their inherent suitability for modelling complex, concurrent, discrete systems, Petri nets have several features specifically advantageous to their application in systems biology:

- They provide an intuitive graphical representation of the system in question;
- There is a large tool support base incorporating numerous analysis techniques and algorithms;
- They provide analysis opportunities at different levels of model completeness (i.e topology through to fully quantified timing parameters);

- There is a large body of literature related to their theory and track record in modelling complex concurrent systems for nearly four decades [PNW04];
- They are readily extendable in a diverse number of ways, for example by the incorporation of time in stochastic Petri nets, or guards in coloured Petri nets.

As discussed in previous sections, there are a large number of different analysis techniques available within the Petri net framework. For ease of description we consider these to be divided into three groups: structural and behavioural properties, timed network behaviour and Boolean based representations. The analysis techniques that can be carried out on these Petri net models are dependent on the information available to the modeller. A current problem in biological modelling is the lack of quantitative data, especially with regard to the kinetic rates of reactions [MS03]. However, if only the topology of a network is known, structural analysis techniques may be applied. If topology and the molecular amounts are known, then both structural and behavioural analysis techniques may be applied. If the topology, initial molecular amounts and kinetic parameters are known, structural, behavioural and timed network analysis techniques may be applied. If the genetic interactions in a regulatory sense are known, Boolean/safe Petri net approaches may be applied. The difference in the view of the network may be viewed more simply as quantitative (behavioural and structural properties) and qualitative (timed network behaviour) [PWM03].

The investigation of biological phenomena was envisaged by C.A Petri as one of the problem areas to which Petri nets could be applied [Pet62], however it is only in the last ten years, with the concurrent advances in biological and bioinformatics techniques that full studies of biological modelling with Petri nets has been carried out. The application of Petri nets in the analysis of biologically systems was pioneered by the work of Reddy *et al.* [RLM96, RML93] (it is often said that Reddy suggested the application of Petri nets to biology [HR04], however C.A. Petri documented this many years beforehand [Pet62]). To date there have been a number of varied studies of the application of Petri nets to biological modelling, and a range of Petri net techniques and extensions have been utilised. These are now discussed.

2.7.1 Behavioural and Structural Analysis

Reddy utilised P/T nets to quantitatively analyse metabolic pathways [RLM96]. Structural and behavioural properties such as P-invariants, T-invariants, liveness and boundedness were analysed and characterised to validate a model of the glycolysis pathway [RLM96]. Reddy *et al.* introduced the first mapping of Petri net elements to biochemical features. In this mapping tokens represent individual molecules, places represent the particular type of a biochemical entity represented by the tokens attached to it, transitions represent discrete biochemical reactions, with the directed arcs of Petri nets indicating the direction that reaction proceeds. This is how most future works mapped biological processes, with some manipulation where necessary. Reddy *et al.* also noted that one of the possible advantages of Petri nets are their extendability, this has proved prophetic as a number of papers have appeared, championing a variety of Petri net extensions [GP98, MDNM00, GKV01].

Hofestädt *et al.* [HT98, Hof94] also investigated the application of Petri nets to the analysis of metabolic networks at a similar time to Reddy *et al.*. Hofestädt *et al.*'s work also suggested the use of Petri nets as a simulation technique. Unfortunately they did not investigate stochastic Petri nets and as a result their simulation framework has been, to some extent, superseded by the theoretical rigour of other simulation techniques available to systems biologists. Kuffner *et al.* [KZL00], utilised Petri nets to store the results of a systematic search of metabolic databases such as KEGG [KGKN02], integrating genomic information and functional annotations. The data was stored in such a way that the future analysis of networks would be facilitated.

Koch, Heiner *et al.* have perhaps been most active in the validation of metabolic networks with Petri nets [KJH05, HKV01, HK04]. The motivation for these studies was the lack of quantitative data hindering the study of these networks via kinetic methods [HKV01]. In their first study they extended the work of Reddy [RLM96] by utilising high level Petri nets to analyse glycolysis and pentose phosphate metabolic pathways in red blood cells. Coloured Petri nets were utilised in order to differentiated the origin and destination of molecules of the same metabolite via their "colour" [HKV01]. Utilising coloured Petri nets allowed the manipulation of the model, removing "unreasonable" processes and cyclic loops. The main results of the paper are then based around detailed T and P invariants analysis, which did not require the use of coloured nets. These invariants were used to validate the model against the information available in the literature. Genrich *et al.* utilised executable coloured Petri nets to investigate metabolic networks, their work enhanced that of Reddy by the use of coloured tokens, these were used to represent both the metabolites name and its concentration [GKV01]. A later paper by Koch, Heiner *et al.* [HK04], again concerned with circumventing simulation techniques due to lack of data, utilises P/T nets rather than coloured nets. In order to exploit T invariants more fully, certain modifications of the pathway were carried out. Here the tokens for all input compounds are seen to be generated by auxiliary input transitions, while all output compounds are consumed by auxiliary output transitions [HK04]. This approach was tested on various metabolic pathways, with the P and T invariant used to validate the derived Petri net models. The most recent contribution by Koch, Heiner *et al.*, again describes Petri nets and metabolic pathways, specifically sucrose breakdown in a potato tuber [KJH05]. They utilise the techniques in the previous paper [HK04], again concentrating on invariant analysis as a method for validating the model, prior to further investigation, for example by kinetic analysis.

2.7.2 Boolean Based Petri Net Representations

The application of Petri nets to metabolic pathways has, as the above studies show, had much success in the early stages of its study. These studies are greatly facilitated by the fact that metabolic networks have a well understood topology, with many of these networks stored in easily accessible databases such as KEGG [KGKN02]. The study of genetic regulatory networks requires a different approach. Often researchers wish to recreate the time series data arising from lab studies. Many authors have pointed out the lack of quantitative data necessary for the simulation of a time course of these networks [KJH05, CRRT04, MS95, EB01], other approaches must be taken to circumvent this problem, for example the automatic parameterisation of these networks [GB01] an approach that is discussed in Chapter 5.

An alternative approach to model gene networks is to view genes in the system as

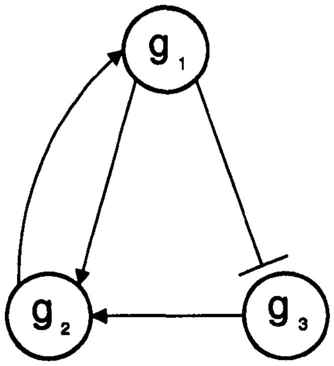


Figure 2.13: A simple Boolean network

abstract switches , having states represented by Boolean values [SBSW05, AKMM98]. There have been a number of studies taking this Boolean view [SBSW05, PS04]. Simplifying a complex regulatory system, encompassing genetic, metabolic and environmental influences into a simple system of Boolean switches has obvious advantages. Figure 2.13 demonstrates a simple gene network, represented as a Boolean model. The Boolean view of genetic regulatory systems has been investigated by a number of groups using the Petri net formalism.

Chaouiya *et al.* investigated the application of Petri nets to Boolean views of genetic regulatory networks, via the use of so called "Boolean Regulatory Petri Nets" (BRPN). [CRRT04]. Techniques such as complementary-place transformation allowed inhibitory relationships to be modelled without undesirable read arcs and ensured that each place is limited to a token capacity of 1. Petri nets where every place has a bound of 1 are know as safe Petri nets. As a result of this bound on token numbers, safe nets are more amenable to state-space based analysis techniques, while still allowing many of the more traditional Petri net analysis methods to be carried out. The drawback of this approach is the large numbers of transitions generated, make simple visual analysis difficult. This approach was then applied to analysis of the *Drosophila* cell cycle and flowering in *Arabidopsis*. The BRPN approach validated the fundamental properties of these networks through the use of simple reachability analysis. Chaouiya *et al.*'s initial investigation [CRRT04] was built upon by Simao *et al.* [SRTC05], by allowing the activity of a gene to be considered at three levels, rather than the pure Boolean views presented previously, more realistic regulatory relations could be investigated. Simao *et al.* attempted to model both the metabolic pathway and the genetic regulation of tryptophan biosythisis. The study focused upon two regulatory feedbacks, namely the inhibition of the enzyme TrpE by the final product of the metabolic pathway, and the transcriptional inhibition of all enzymes by the holorepressor, involving Trp at high concentrations. They validated their modelling technique, finding that the network was structurally bounded (that is each place is bounded regardless of the initial marking) and conservative (a net with k initial tokens remains k-bounded) which are to be expected from a safe net system. Upon investigation of the reachability of system states they found their model successfully replicated the dynamics of homoeostatic levels of internal trytophan and TrpE activity under low levels of external Trp [SRTC05]. Under moderate Trp influx they found a dead marking with TrpE and the repressor inactive. Finally under high external Trp 6 dead markings were discovered, interpreted as illustrating that the metabolic pathway can be inhibited at any of its six steps.

An alternative approach into the Boolean analysis of regulatory network was put forward by Steggles *et al.*. This method has a number of differences to that of Chaouiya *et al.* [CRRT04], firstly logical simplification techniques, in particular the McCluskey minimisation algorithm [Bre92], were utilised, greatly reducing the size and complexity of the resulting network. Secondly a two phase commit protocol is utilised to aid the simulation and analysis of the network. The technique was then applied to the analysis of sporulation in *Bacillus subtilis*, with reachability analysis utilised to validate the fundamental regulatory properties of the model [SBSW05].

2.7.3 Timed Network Analysis

There are a number of ways of associating time with a Petri net in order to investigate the time evolution of a biological system. Timed Petri nets can offer a deterministic approach [PWM03, Mur89], but have remained unused presumably due to the mature ODE tool support available to systems biologists. While Petri nets may not provide an especially usefully way of simulating deterministic systems it should be noted that Petri nets may be used to perform other, model checking analysis on networks whose time evolution is typically evaluated using ODEs [HKV01, HK04]. In the sequel we examine modelling attempts utilising hybrid and stochastic Petri nets.

Stochastic Petri nets (SPNs) [Mol82], extend P/T nets by adding a firing rate to each transition. Stochastic Petri nets are Markovian systems operating in continuous time [Mar89] and as such have a strong theoretical justification for their use in biochemical systems [Gil77]. Indeed it is though that stochastic simulation techniques are required to capture fine grained network behaviour and pathway switching, that is crucial to the analysis of regulatory systems [MS03, SPB01]. While there are some inherent problems with the stochastic simulation of biochemical networks [EB01, MS03] stochastic Petri nets have been used in a number of studies to date. The earliest investigation into the applicability of SPNs to the modelling of biological networks was that of Goss and Peccoud [GP98]. They utilised the SPN tool "Mobius" [CCD$^+$01] to model and simulate the stabilising effect of the ROM protein on the replication of the Col-El plasmid. Their study utilised the SPN formalism to find distributions of protein numbers, plasmid numbers time courses. The study showed the applicability of SPNs to molecular systems with "small" numbers of reactants, however the results gave no greater insight when compared to the mean on an ODE approach [GP98]. The authors also called for optimisation techniques, required by the lack of quantitative kinetic data, which has become a current research topic in Systems biology. A study conducted at a similar time to that of Goss and Peccoud, was the use of Stochastic Activity Networks (SANs), again modelled inside the Mobius tool [CCD$^+$01], in the analysis of blood coagulation in pathological states haemophilia A and B [ML97]. SANs are very similar to SPNs, the main difference between the two is the use of "gates" to control the input and/or output of certain transitions. The interesting work carried out here was the "divide and conquer" approach where smaller networks are refined against existing data, and then a number of these networks integrated into the final, composed model. The simulation data from the full model successfully reproduced the biological results. The same group went on to simulate the pathways involved in human menopause [TL02]. This study utilised similar techniques and tools, e.g Mobius and SANs [CCD$^+$01]. Again small subsystems were validated against experimental data before the smaller networks were integrated, again the simulation data matched well against the available experimental data. SPNs have also proved to provide an insight into bacterial regulatory networks. Srivastava *et al.* utilised SPNs to investigate regulatory pathways in *E. coli*, specifically the response of the sigma factor, σ^{32} to heat and ethanol shock [SPB01]. The time courses were analysed and correlated well to lab data, interestingly the simulation results gave a useful insight into the end locations of the sigma factor in relation to the chaperon proteins.

An alternative to the use of SPNs is the use of hybrid Petri nets [Heb70]. Matsuno *et al.* have been productive over recent years, providing a hybrid Petri net architecture for simulating biological processes [NDMM04, MDNM00, MFD+03]. These nets are referred to as Hybrid Functional Petri Nets (HFPN) [NDMM04]. HFPN were developed due to perceived inadequacies with ODE based techniques that made them inaccessible to biological researchers due to poor GUI interfaces [NDMM04, MDNM00]. While hybrid Petri nets have great potential in the integration of metabolic and regulatory networks, it is unclear from Matsuno *et al.*'s papers what theoretical justifications have been considered in relation to their hybrid approach. . The tools produced by the group have been made available as commercial packages, this may hint at why there is little in-depth insight into theoretical assumptions and algorithms utilised in their approach.

Other useful investigations into Petri nets and biology include Schuster *et al.*'s investigation into the decomposition and analysis of metabolic networks from *Mycoplasma pneumoniae* [SPM+02]. The authors note that some of the analysis techniques developed were equivalent to methods already available in the Petri net literature. Finally Pelag *et al.* surveyed many analysis and knowledge capture methods and proposed that a mixture of Petri nets and workflows captures knowledge and relations well and is amenable to analysis. They suggest that this technique may be extended utilising kinetic methods or Bayesian reasoning techniques.

At present, Petri net models described in the literature, (both biological and non biological) have generally been encoded by hand, using expert knowledge. This is a time consuming and error prone process that may involve the duplication of models that have already been defined in a different modelling environment. The advent of common modelling languages for both Petri nets and systems biology, discussed in Section 2.8, open up the possibility of automatically generate Petri net models from predefined biological networks in repositories such as KEGG [KGKN02]. This is discussed further in Section 3.3.

2.8 XML Interchange Formats

The discipline of Petri net research and the emerging field of systems biology have numerous tools and techniques for modelling complex concurrent systems. The storage and interchange of models is an important aspect of modelling in any domain [IWK$^+$04]. As such, model storage and interchange techniques and formats are crucial to the development of the fields of Petri nets and their application within systems biology. In order to analyse and recreate previous model based studies is is imperative that the models be fully published and accessible to a range of computational tools. The publication of a model used in scientific literature may prove problematic where the model is defined in a formalism that is not amenable to plain English interpretation, or if the model is simply too large to fit within the space of a scientific paper. Assuming the model is fully published, how are other researchers able to evaluate and learn from it? The analysis of a model via a computational tool has traditionally been a problem in both Petri net and systems biology related disciplines, due to the variety of tools, and the accompanying variation in (or lack of) standards. Previous tools simply used a proprietary text or binary file format for saving their results meaning that models had to be rebuilt completely if researchers needed to evaluate them. Models also had to be frequently rebuilt due to upgrades in a particular tool [HF$^+$03]. In addition to being able to share models, modelling disciplines need a well developed repository of test models and software tools with which to evaluate these. The development of a suitable test suite facilitates tool validation, while providing an important educational resource for researchers.

A number of eXtensible Markup Language (XML) [W3C04] based standards have been developed and have begun to facilitate the interchange of models. In this section the Systems Biology Markup Language (SBML) [HF$^+$03], the Petri Net Markup Language (PNML) [WK02] and CellML [LHN04] are discussed. The Systems Biology Markup Language was envisaged by a large and wide ranging consortium of systems biologists in 2000 and level 1, which contained a definition of a number of elements needed to produce large scale models, was officially released in 2002 [HF$^+$03]. SBML aims to become the standard language of computational systems biology [Kit02] and is fast becoming the *lingua franca* of systems biology. Around the SBML standard has sprung a wide range of tools and aids from the community. A number of tools, both deterministic and stochastic have been developed to utilise SBML models. Many of these tools can be found via links from the SBML website, including tools such as Cellware [DMS$^+$05] and Gepasi [MK98], there are also tools to facilitate the study of biological networks, such as network editors and automatic layout tools. There are also attached to the site a test suite of models, for example to evaluate a new tools performance, and a community support and information forum.

A SBML model is made up of the following components:

- **Compartments**, containers of a finite volume where reactions take place;
- **Species**, a chemical substance or entity that takes part in a reaction;
- **Reaction**, a transformation, transport or binding process that can change one or more species;
- **Parameter**, a quantity with a symbolic name;
- **Unit definition**, a unit used in the expression of quantities in a model;
- **Rule**, a mathematical expression constructed from the set of reactions, rules can establish constraints, set parameters etc.

A great deal of modelling power can be realised using the components above, although there are constant improvements being introduced by the community, especially in the the area of stochastic simulation.

The Petri Net Markup Language [WK02] has an almost identical aims and objectives to SBML. Nearly forty years of Petri net research has led to a large tool support. Unfortunately, as with the early systems biology tools, they relied on proprietary file formats for model storage. Many Petri net tools allow investigation into specific, specialist aspects of Petri net theory. In order get the most out of the analysis of a model, it is necessary to utilise the specialities of a number of different tools. PNML was designed to facilitate the interchange of data between Petri net tools by adhering to the following principles:

- **Readability**, XML is employed to facilitate computer readability but the XML must be understood by human eye
- **Universality**, The format should allow any version of Petri nets, with any extensions to be represented
- **Mutuality**, The format should allow as must information to be obtained as possible, even if the Petri net type is unknown to the tool, i.e it should be possible to extract the common principles and features of Petri nets.

A PNML model is comprised of the following features:

- **Petri net objects**, most commonly places, transitions and arcs, but also includes paging information;
- **Labels**, each object may have a set of labels, for example, a place name or marking;
- **Graphical information**, this states where the object will appear on the screen;
- **Tool specific information**, Allows any arbitrary information not covered by objects and labels. Other tools can safely ignore this;
- **Page and reference nodes**, a page can consist of other objects and are placed together by the use of reference nodes.

The Petri net markup language also presents some potentially fruitful ideas regarding the composition of models [KW01b]. Models can build recursively from module instances allowing a system to be composed from small modules and allows module instances to be created from a common template. This feature is potentially very applicable to modelling biological network enabling small systems, for example translation, to appear multiple times in a large module, with a small alteration, for example the gene being transcribed. There has been some discussion on extending PNML to include biochemical information via the creation of a biological Petri net markup language [CFKR02], however with the community adoption of PNML and SBML it is considered that the conversion between formats (for example [SKSW04a]) is currently a more desirable option than multiple specialist formats.

CellML [LHN04] is a XML based, open standard for the interchange of biological models. CellML aims to allow researchers to share models, and to reuse components from one model in another, thus accelerating future model building. CellML models include information about model structure, mathematical rules, and metadata, which help researchers to query specific model components in databases. CellML includes a model repository and has a large tool support. CellML clearly shares many attributes and aims with SBML. Currently many components are amenable to SBML conversion (and vice versa), with a number of converters available. The CellML and SBML development teams are actively discussing how the two languages can be made to work together, however, it is felt that SBML and CellML have sufficiently different aims to justify the two formats [Cel06]. For example, CellML version 2 will allow the use of ontologies to document and validate models.

Chapter 3

Place Transition and Safe Petri Nets: Import and Analysis

3.1 Introduction

One of the advantages of the Petri net framework is that many interesting properties of a network can be gleaned with a relatively small amount of knowledge about the system. If the connectivity of the network is known then structural properties can be analysed. Analysis of these structural properties allow researchers, for example, to identify the conservation of post-translational modification to proteins along a signal transduction pathway or feedback loops in a regulatory pathway using invariant analysis [RLM96]. If, along with the connectivity of the network, some initial state (quantity of each molecule) is known then behavioural Petri net properties can be analysed. This allows us to ask questions such as "is state *A* reachable from state *B*?", "will reaction *X* ever occur under these conditions?" and "will this protein accumulate?".

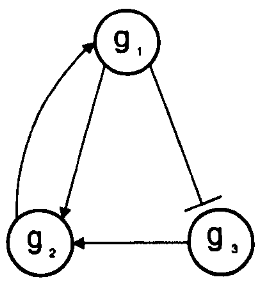
Asking questions such as these within the formalism of Petri nets [Mur89] can validate our understanding of a system and create new testable hypotheses for laboratory experiments. Validation of a model verifies that the model in question satisfies our understanding of the real life properties of that system, for example in air traffic control, checking that there are no collisions. Validation of models is a key step for further investigation into the models properties, and it is argued by Heiner and Koch that in systems biology, model validation of structural network properties is an important first step before the qualitative simulation of these models [HK04]. With the gaps in biological knowledge, validation would serve to highlight gaps in our understanding of the system. For example if some properties of the model do not match results obtained from the lab, then there are most likely some shortcomings in the model or the original data that must be rectified. Analysis of the model may also reveal interesting emergent properties that were not considered in the modelling process, but point to interesting behaviours that may not be currently unexplained. Structural and behavioural Petri net analysis may provide testable lab experiments, allowing lab workers to test sensible (according to the model) hypotheses, saving both lab time and money. In this chapter the investigation into the behavioural and structural properties of biochemical networks is discussed in two sections. In the first section the issue of modelling genetic regulatory networks is discussed. Here analysable models can be constructed and investigated based on the limited knowledge available about some regulatory interaction between genes. The second section investigates the analysis of metabolic networks where the topology is well known. A method is presented allowing the automatic import of metabolic models (or any model encoded in SBML [HF$^+$03]) into the Petri net framework, via PNML [BCvH$^+$03]. A number of metabolic networks imported from both the systems biology literature and the KEGG database [KGKN02] then have their structural properties investigated.

Genetic regulatory networks are inherently complex and more levels of complexity are being uncovered, such as reverse transcripts (transcribing RNA into DNA) and epigenetics (changes in gene regulation that occur without a change in the DNA sequence) [SA00]. To facilitate the preliminary investigation of these complex networks it is necessary to make a number of simplifying assumptions. These assumptions allow an initial investigation into these systems, with the hope of identifying knowledge gaps, hypothesis generation and further understanding of the genetic system. One possible assumption that can simplify the analysis of these systems is to view genetic regulatory networks in the binary domain. This approach, pioneered by Kaufman [Kau69], views genes as Boolean switches that can either be "on" (the gene is expressed) or "off" (the gene is not expressed). Genes have causal relationships, allowing genes to activate or inhibit each other, either individually or in small clusters. It is immediately apparent that this point of view ignores subtleties such as basal expression levels and response governed by differential amounts of transcript. Viewing genetic regulatory networks as Boolean based systems does have a number of advantages, such as computational efficiency and robustness to noise [SBSW05]. Thus these studies can still be of great use in preliminary studies. This Boolean assumption has been investigated utilising a number of modelling formalisms, including Boolean networks [AKMM98], Bayesian networks [MM99] and Petri nets [SBSW05]. In order to alleviate the inherent state space analysis problems (where multiple Boolean states combine combinatorially), this chapter utilises a Petri net based approach described by Steggles *et al.* [SBSW05]. This approach converts Boolean models into Petri nets, allowing a model to be subject to a number of analysis techniques that can cope with large state spaces. This approach is outlined and then utilised on a novel Boolean model of the classical Lac operon in *E. coli.* The resulting models are analysed within the Petri net framework using a number of Petri net tools. The model has its behaviour successfully validated, replicating the high level behaviour described in the literature. The techniques are taken forward to construct a model of phosphate stress response of *B. subtilis* as part of a larger case study in Chapter 6.

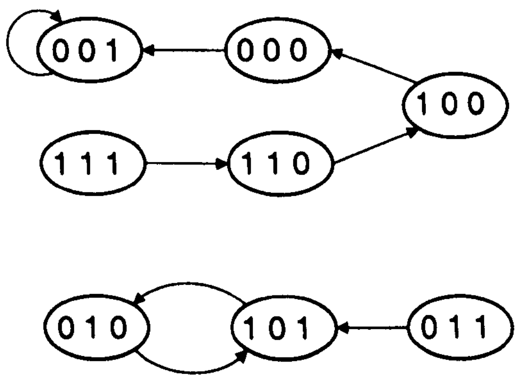
If the topology of a network or pathway is well known, Petri net structural analysis can be applied [Mur89]. Metabolic networks typically have a well understood topology and are available in databases such as KEGG [KGKN02]. These databases are comprehensive and provide a valuable resource to researchers. To effectively utilise resources such as these there must be ways of automatically translating this information into a model suitable for analysis. It would be inconceivable for the complete proteomic, metabolomic and genomic networks of an organism to be constructed and interpreted manually. Presently, there are many readily available data sets and model repositories such as KEGG [KGKN02]. In order to effectively utilise these resources, there must also be ways of transforming this data in to a form ready for analysis with Petri nets. With the Systems Biology Markup Language (SBML) [HF$^+$03], fast becoming the *lingua franca* of systems biology, it would be advantageous to be able to automatically convert SBML files into a Petri net format, allowing the import of a large number of models ready for analysis with Petri net tools. The Petri net community have introduced an interchange format, PNML (Petri Net Markup Language) [BCvH$^+$03], which is an ideal candidate for the Petri net format. In this chapter a mapping is presented allowing the automatic translation of models from SBML to PNML. This mapping forms the basis for a prototype tool implemented using XML [W3C04] and Java technologies. This tool is then utilised to import a metabolic pathway, allowing structural analysis with the Petri net tool INA [Sta04].

3.2 Petri Net Representations of Boolean Based Genetic Networks

A comprehensive, fully quantitative modelling based study of genetic regulatory networks requires a large amount of knowledge of the system, as well as analysis techniques. Network topology, initial concentrations, kinetic rate constants and environmental conditions all have to be factored into the model, with this information generally being unavailable for the complete model [HK04, EB01]. These problems can be mitigated to some extent by the parameter inference or estimation techniques discussed in chapter 5. The study of genetic regulatory networks can be greatly simplified by viewing gene networks as Boolean based systems [AKMM98], where genes represent abstract binary switches [SBSW05] which can be either "on" or "off". These genes have causal relationships between them, where genes can inhibit or activate one another. This approach was pioneered by Kaufman [Kau69] and has formed the basis of a number of studies. Figure 3.1(a) shows a simple model taking this Boolean view (taken from [AKMM98]). In this model three genes interact: g_1 and g_3 combine to promote the expression of g_2; g_1 inhibits the expression of g_3; and g_2 promotes the expression of g_1. Systems such as these can be investigated by the use of state transition graphs where the whole state space of the system is explored. The state space of the model in Figure 3.1(a) is depicted in Figure 3.1(b).



(a) A Boolean network with three genes



(b) A state transition graph of the Boolean network in a)

Current State			Next State		
g_1	g_2	g_3	g_1'	g_2'	g_3'
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	0

(c) The truth tables relating to a)

Figure 3.1: An example of a Boolean network with three genes and the resulting state transition graph

Boolean based models are promising for the study of genetic networks due to the suitability of the abstraction to available biological data [AKMM98]. However they typically suffer from a number of problems, namely coping with very large state spaces, and an inability to cope with inconsistent or incomplete data [SBSW05]. These problems can be alleviated by translating the models into Petri nets. The transformation of Boolean based biological models into Petri nets was initially carried out by Chaouiya *et al.* [SRTC05], and further extended by Steggles *et al.* [SBSW05]. The approach of Steggles *et al.* further address the fundamental problem of state space explosion by first simplifying the model using well studied logic reduction techniques. These compact logical relationships are then automatically translated into Petri net structures by a bespoke tool, GNaPN (Gene Networks as Petri Nets) [GNa06]. GNaPN allows both the asynchronous and synchronous semantics of genetic regulatory networks to be modelled and analysed. The methods presented by Steggles *et al.*, discussed in the next section, are used to reverse engineer and analyse a Boolean abstraction of the well studied Lac operon [JM61]. Later in chapter 6 a model of the phosphate stress response in *Bacillus subtilis* [BVV+97, DLY02, DDB+04]. Here a model of the Lac operon is successfully validated by state space analysis techniques. Proving the model is consistent with current knowledge of the system at a Boolean level gives great confidence to utilise other techniques and assist in the creation of more complex models, for example utilising stochastic Petri nets.

3.2.1 Logic Simplification and Gene Networks

In the field of microprocessor design, a technique called logic simplification is used to optimise digital circuits built using Boolean logic. Logic simplification can be used to simplify Boolean representations of biological networks [SBSW05], utilising techniques such as the Quine-McCluskey minimisation algorithm [Bre92]. Logic reduction not only assists in the creation of Boolean models, but also greatly aids in the analysis of such systems.

To demonstrate how logic simplification can be applied, consider the simple Boolean network presented in Figure 3.1(a), and the accompanying truth table in Table 3.1(c). From the truth table in Table 3.1(c), each gene can be represented by Boolean terms that fully describe each gene's behaviour using the techniques presented in [SBSW05]. These Boolean terms are made up of a number of "minterms". These minterms represent system states in which the next state results in the gene being on. The Boolean summation of these terms completely describes the behaviour of the gene. For example, the gene g_2 can be described by the following Boolean term:

$$g_2 = (g_1 \wedge \overline{g_2} \wedge g_3) \vee (g_1 \wedge g_2 \wedge g_3), \tag{3.1}$$

where $\wedge$ represents logical "and", $\vee$ represents logical "or" and hence $g_1 \wedge \overline{g_2} \wedge g_3$ represents the state 101. The Boolean terms evaluated from the truth tables are often overly complex, holding much redundant information [SBSW05]. Reducing the size of these terms is extremely beneficial for future analysis, especially with state space based methods. The Quine-McCluskey algorithm [Bre92] minimises logic terms by merging miniterms that differ by only one variable. For example, consider the term in Equation 3.1, this has two miniterms that differ by only one variable, g_2. It is straightforward to see how this term can be simplified by the removal of this variable, and merged into the simpler term $g_1 \wedge g_3$, which is logically equivalent to the original term, and is in fact equivalent to the original model shown in Figure 3.1(a).

The aim of the approach detailed above is to be able to reverse engineer the underlying network from an understanding of the Boolean behaviour of the network. While the complete truth table is an ideal starting point from which to reverse engineer the underlying Boolean model, it is generally unavailable. Instead a subset of the complete truth table is utilised. In the main a good estimate of the complete Boolean behaviour can be gleaned by the description of genes and their nearest neighbours. This information can be gained from both expert knowledge and literature searches. These Boolean terms can then be taken forward to generate an investigatable Petri net model, this process is now discussed.

3.2.2 From Boolean Logic to Petri Nets

Petri nets have been proposed as a formalism in which to study Boolean networks. Petri nets have a large body of tools and techniques and can alleviate problems of state space exploration and can cope with incomplete or inconsistent data [SBSW05]. Petri nets [Mur89], have a large range of analysis techniques available too them, with 1-bounded [Mur89] or safe Petri nets having tractable techniques to evaluate state spaces, such as unfolding [Kho03]. The use of Petri nets to model Boolean based genetic networks was originally presented by Chaouiya [CRRT04]. Chaouiya *et al.'s* used complimentary places to model

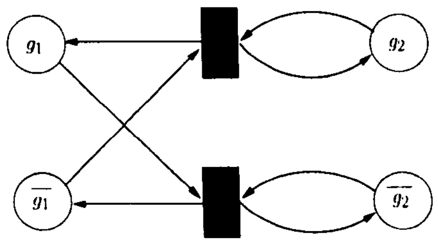


Figure 3.2: A Petri net representation of g_2 activating g_1, from [CRRT04]

an entities state being either on or off. Chaouiya *et al.'s* technique produced asynchronous Petri nets and has been tested on a number of small models [CRRT04]. This method has been advanced by the work of Steggles *et al.* [SBSW05], as discussed this method reduces the complexity of a network by the use of logic reduction techniques. The method of Steggles *et al.* also provides a tool support allowing the automated construction of these networks via a tool, GNaPN which allows the export of either asynchronous nets or nets following a synchronous commit protocol. In this work the asynchronous nets are utilised. Steggles *et al* also considered synchronous nets, with an update phases, however these are not utilised and are not discussed further. The construction of asynchronous Petri nets from Boolean logic is now discussed.

The idea behind both strategies is to model each gene g_i with two complementary places g_i, representing the gene being "on" and $\overline{g_i}$, which represents the gene being "off". With careful construction, these complementary places remove the need for inhibitor arcs, and an asynchronous model can be constructed [CRRT04]. An example of this is shown in Figure 3.2. This figure describes the simple relationship of g_2 promoting g_1. The idea here is for a gene, g_i to be on, there must be a token on P_i, and the input transition for P_i must have an input arc from $\overline{P_i}$, any genes that promote g_i and the complementary places of any genes

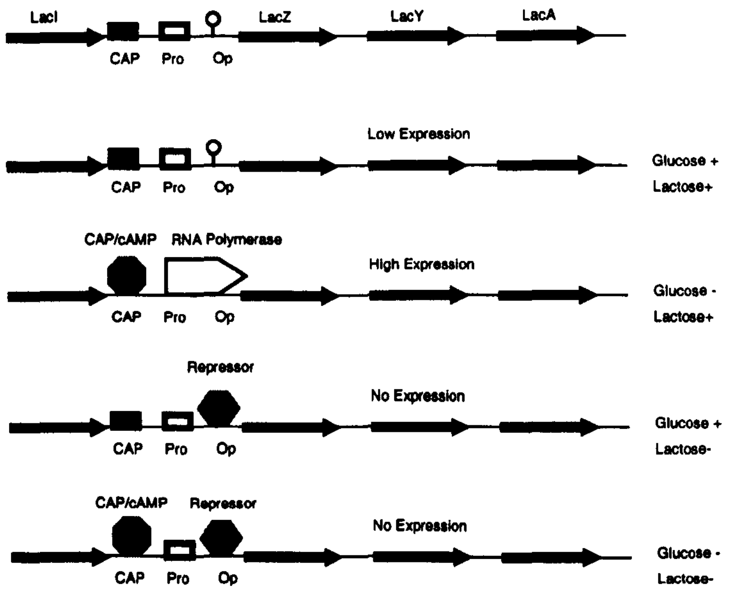


Figure 3.3: The Lac operon in *E. coli* [JM61]. Here the proteins bind in a number of combinations on the promoter, effecting the resulting expression of LacZ, allowing the bacterium to only utilise a lactose resource in the absence of glucose.

that inhibit g_i. It is straight forward to see how this approach can be utilised to construct larger networks.

3.2.3 Modelling and Analysis of The Lac Operon

In order to facilitate the discussion of the application of the modelling approach presented by Steggles *et al.* the classical model of the Lac operon in *E. coli* is utilised [JM61]. The Lac operon is a classic regulatory system. The Lac operon allows an *E. coli* to adjust its metabolism depending on whether glucose, lactose or both glucose and lactose are available. *E. coli's* primary source of food is glucose. This does not have to be modified before it is utilised by the bacterium's respiratory pathway. Lactose must go through a number of conversion steps before it may be utilised. This means that lactose is a more costly form

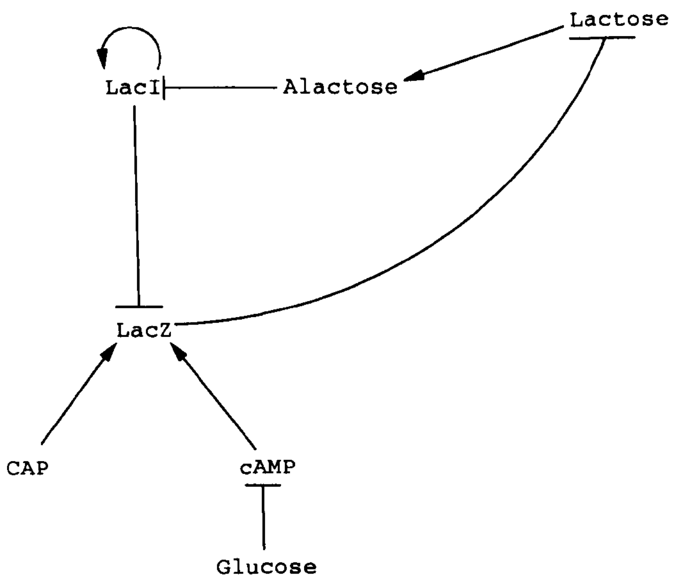


Figure 3.4: A Boolean model of the Lac operon. This depiction is then utilised to describe its truth tables

of energy, and glucose will be used exclusively even if lactose is available [JM61]. As the lactose machinery is expensive to maintain, it is switched off in the presence of glucose in order to save energy. The Lac operon is pictured in Figure 3.3. When there is not lactose in the system a repressor protein, LacI binds to the Lac operon's promoter. This prevents RNA polymerase from binding and transcribing the operon. When lactose is present, an isomer of lactose, alactose, binds to the repressor, forming a complex. This prevents the repressor from binding to the promotor and allows RNA polymerase to bind and transcribe the genes in the operon. Under conditions where both glucose and lactose are available, the lactose machinery is only transcribed at a basal level, even though the repressor has been rendered inactive. This is due to the effect of the catabolite activator protein (CAP). CAP can only bind to its DNA binding site near the promotor in the presence of cAMP. If glucose is present the level of cAMP is low, so CAP does not bind to the DNA and hence does not activate the RNA polymerase. When glucose levels are low the cAMP level rises,

LacZ	lactose	Lac
0	0	0
0	1	1
1	0	0
1	1	0

(a)

lactose	alactose
1	1
0	0

(b)

alactose	LacI
1	0
0	1

(c)

LacI	cAMP	CAP	LacZ
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

(d)

glucose	cAMP
0	1
1	0

(e)

Figure 3.5: Truth tables describing the behaviour of the Lac operon. The next state of the species of interest is shown on the right, relative to the current state of its nearest neighbours

allowing CAP to bind, activating the transcription of the Lac operon.

The molecular species listed in the model are the proteins LacI and LacZ, the metabolites glucose and lactose, and the activity regulators cAMP and CAP. For simplification, LacZ is the only protein transcribed from the Lac operon that is considered in the model. LacY and LacA are omitted from the model as their behaviour is the same as that of LacZ. It is noted that there has been a simplifying assumption made to allow the Lac operon to be modelled by the Boolean based Petri net modelling technique proposed by Steggles *et al.*. As discussed previously, in the presence of both lactose and glucose, LacZ is expressed at basal levels. This basal expression does not fit with the Boolean modelling technique and is simplified by viewing the basal expression as the gene being off.

The model can be seen in Figure 3.4. The items of interest in the model are then listed, along with the affecting molecules in the immediate neighbourhood. The next state of each molecule is listed relative to the current state of its nearest neighbours. This information is captured in a series of truth tables as listed in Figure 3.5. These truth tables are then converted in to the following series of Boolean equations:

$$lactose = (\overline{LacZ} \wedge lactose) \tag{3.2}$$

$$alactose = lactose \tag{3.3}$$

$$LacI = \overline{Alactose} \tag{3.4}$$

$$LacZ = \overline{LacI} \wedge cAMP \wedge cap \tag{3.5}$$

$$cAMP = \overline{glucose} \tag{3.6}$$

The Boolean equations are then minimised using the approach outlined in Steggles *et al.* [SBSW05], via a Java tool written by one of the authors (R.Banks). The tool, GNaPN, minimises the truth tables and automatically produces an asynchronous Petri net according to the rules and techniques described above. The resulting Petri net model was then exported to the PNML [BCvH$^+$03] language, to allow the model to be analysed using PNML compliant tools.

The resulting model then has its state space investigated using reachability analysis. Reachability analysis is described in chapter 2 and involves the systematic and efficient exploration of a Petri nets possible set of markings dependant upon an initial marking. The reachability analysis was carried out using the PEP tool [Gra97]. Reachability is a marking dependent property of a Petri net and hence initial markings had to be set. The places of interest in the resulting net are the places indicating the presence or absence of a particular gene or condition. For example, for the protein LacZ there exist two complementary places, *LacZ* and $\overline{LacZ}$ indicating the presence or absence of LacZ respectively. For these places, the initial conditions used were, *LacI+, cAMP+, LacZ- and CAP+*. The presence of glucose and lactose was then considered in all four possible combinations.

The net was then assessed for reachability of certain states in PEP [Gra97]. In particular the model was assessed as to whether the individual places of *LacI+*, *LacZ−* and *alactose+* ever became marked. The results, and their implications were as follows.

- **Alactose** The presence of Alactose is an indication of the presence of lactose. Reachability analysis showed that it was possible to have a state where alactose is present, only in the presence of lactose.

- **LacI-** The absence of LacI indicates that lactose is present in the system and has inhibited the expression of LacI via the presence of Alactose. Reachability analysis showed that LacI could only become absent in the presence of lactose.

- **LacZ+** The presence of LacZ is a key indicator of the operon, and indicates that *E. coli* has resorted to the lactose pathway. Reachability analysis showed that LacZ was only present when both glucose was absent and lactose was present.

All of these results were in agreement with the current biological understanding of the system and assumptions used in the model. This demonstrates that the technique appears suitable for the modelling of Boolean representations of genetic regulatory networks. Importantly, the state space needed for the analysis was surprisingly small with reachable

Initial Marking		Reachable Marking		
Glucose	Lactose	Alactose	LacZ	LacI-
√	√	√	√	√
√	X	X	X	X
X	√	√	X	√
X	X	√	X	X

Table 3.1: The reachability results obtained via analysis of the Lac model within the PEP tool

states numbering in the order of 100. Moreover, the technique proved to be extremely efficient. All the analysis carried out on this model took less than a second on a 1gHz machine.

3.3 Import and Analysis of Metabolic Networks

As discussed in Section 2.8, model documentation and interchange is a important aspect of systems modelling. This has become apparent in both the disciplines of systems biology and Petri net research, with the development of SBML [HF$^+$03] and PNML [BCvH$^+$03] respectively. Associated with these community accepted standards are a number of example models. SBML in particular has a test suite, and a number of models published in scientific journals. A repository of these is available at the SBML website [SBM04]. There are also a number of other network repositories that have become available as SBML models. A prime example is that of the KEGG metabolic pathways database [KGKN02]. Here, a large number of metabolic pathways are available to researchers. As discussed previously, Petri nets have begun to be applied to analysis in systems biology. In order to fully exploit the application of Petri nets to systems biology it is important that a researcher can import existing SBML models into the Petri net framework, enabling their analysis via a number of PNML compliant tools such as PEP [Gra97] and the Petri net Kernel [PNK04].

In the remainder of this chapter, a novel mapping of biochemical networks (represented in SBML) to Petri nets (represented in PNML), is considered in detail. Initially, the direct mapping of a SBML model into a PNML description of a P/T net is described. This mapping forms the basis for a prototype tool, implemented in Java, that converts SBML level 1 files into PNML. This tool is then utilised to import a SBML model of a glycolysis pathway from the KEGG database into a PNML file. The imported model is then analysed with the INA tool, demonstrating the immediate accessibility of the resulting imported models to Petri net analysis. Finally, to demonstrate the large scale effectiveness of the conversion tool, all pathways from KEGG relating to *B. subtilis* are imported into PNML.

Alternative ways of modelling biological systems with Petri nets are then discussed, with particular attention paid to how the SBML/PNML place/transition net mapping can be extended to produce stochastic Petri net models. Work into this area requires current PNML standards to mature before this application can be completed.

3.3.1 Mapping from SBML Level 1 to a P/T Net encoded in PNML

To enable the automatic import of SBML models into PNML, it is necessary to establish a mapping of the underlying biological phenomena to the Petri net framework. The work of Reddy [RLM96] was pioneering in this respect, and provides an excellent starting point for this work. In this approach, biological pathways are viewed as a series of parallel discrete events. One molecule of a biochemical substance is deemed equivalent to a single token [RLM96] in the accompanying Petri net. Generally, transitions are equivalent to biochemical reactions, with arcs denoting the direction of the reactants. This simple approach is now further defined, demonstrating how a SBML model can be mapped to a PNML representation of a P/T net.

Each SBML file contains a list of species which take part in the biochemical reactions of a system. Each species is specified using a `<species>` field which contains a unique `<ID>` attribute to name the species. A `<compartment>` field identifies where the the given species is found; in stochastic systems an `<initialAmmount>` field is also required. Each species will be represented by a place in our Petri net model, and so is mapped to a PNML `<place>` field. The species `<ID>` attribute then becomes the place `<ID>` attribute and `<name>` field. For example the following PNML would be used to a represent a glucose species in a SBML model.

```
<place ID="glucose">
<name>
<text>"glucose"</text>
</name>
<initialMarking>
<text>"0"</text>
</initialMarking>
</place>
```

One added complication is that a species may occur in more than one compartment in an SBML model. In this case it is clear why the `<ID>` attribute must be used as the identifier. The information about the compartment may be retained as part of the modular or graphical elements. The initial concentration or the initial amount field will be used to

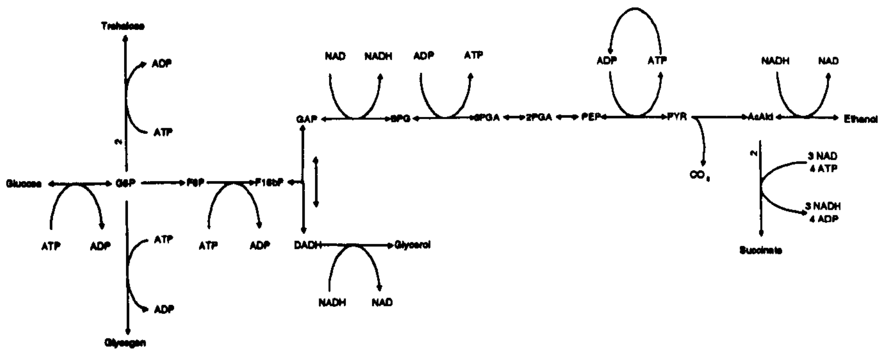


Figure 3.6: The *Saccharomyces cerevisiae* glycolysis pathway from [TPR$^+$00]. Abbreviations: G6P, glucose 6-phosphate; F6P, fructose 6-phosphate; F1,6bP2, fructose-1,6-bisphosphate; GAP, d-glyceraldehyde-3-phosphate; BPG, 1,3-bisphosphoglycerate; 3PGA, 3-phosphoglycerate; 2PGA, 2-phosphoglycerate; PEP, phospho-enol-pyruvate; PYR, pyruvate; AcAld, acetaldehyde; DADH, glycerone phosphate; ATP, Adenosine triphosphate; ADP, Adenosine diphosphate; NAD, nicotinamide adenine dinucleotide (oxidised); NADH, nicotinamide adenine dinucleotide (reduced).

calculate the Petri nets initial marking and this is discussed in more detail below.

In an SBML file the `<reaction>` field is used to specify a single reaction that takes certain species as inputs, as specified by the `<listOfReactions>` field, and produces certain species as products, specified by the `<listOfProducts>` field. Each reaction is represented by a transition in the Petri net model using the PNML `<transition>` field and the unique `<ID>` attribute associated with each SBML reaction maps to the `<ID>` attribute of the corresponding transition. Arcs then need to be defined that will be used to connect the input/output places to this PNML transition. Note that in SBML there is no concept similar to an arc. To solve this, we create an `<arc>` entry from each species place in the list of reactants to the new transition and from the transition to each species place in the list of products. Each arc, A_i, is given an `<ID>` i where i is an integer that relates to the order in which the arcs are created.

Figure 3.7 shows how the reaction shown in Figure 3.6 is mapped to a transition in

```
- <reaction name=" vTreha " reversible=" false ">
- <listOfReactants>
<specieReference specie=" G6P " stoichiometry=" 2" />
<specieReference specie=" ATP " stoichiometry=" 1" />
</listOfReactants>
- <listOfProducts>
<specieReference specie=" Trh " stoichiometry=" 1" />
<specieReference specie=" ADP " stoichiometry=" 1" />
</listOfProducts>
......
</reaction>
```

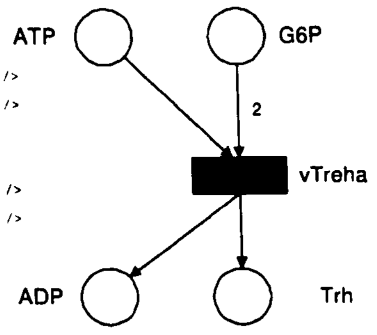
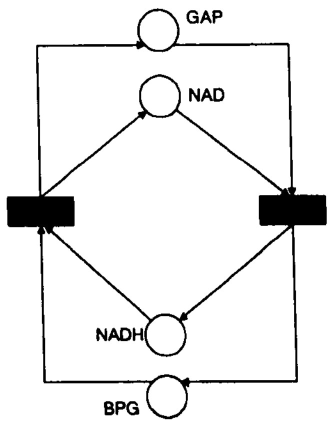


Figure 3.7: Creation of a simple transition from SBML.



```
- <reaction name=VGAPDH" reversible=true">
- <listOfReactants>
<specieReference specie=GAP" stoichiometry=1" />
<specieReference specie=NAD" stoichiometry=1" />
</listOfReactants>
- <listOfProducts>
<specieReference specie=BPG" stoichiometry=1" />
<specieReference specie=NADH" stoichiometry=1" />
</listOfProducts>
.....
</reaction>
```

Figure 3.8: How a reversible reaction is mapped

PNML. A reaction tag may have the `<reversible>` field set to true to indicate that the reaction can also occur in the reverse direction. If this is the case then we create a second transition for the reaction which we name `ID_R` (where ID is the original reaction `<ID>`) such that the input/output places of the original transition are reversed. An example of such a Petri net for a reversible reaction and the associated SBML fragment is given in Figure 3.8.

Given the Petri net structure it is now necessary to derive the initial marking for the PNML model. We set the `<initialMarking>` attribute in PNML using either the SBML species `<initialAmount>` or `<initialConcentration>` fields, resulting from stochastic or deterministic models respectively. The <initialAmount> field defaults to define the initial quantity of the species as an absolute amount [HF$^+$03]. If, however, a <substanceUnits> field is given, then <initialAmount> is defined in terms of that unit. If the value is in a molar form then the <initialMarking> can be obtained simply by multiplication by Avogadro's number [GP98]. If the <substanceUnits> are a multiplication of a molar amount, then a simple conversion can be carried out to obtain the molecular amount. In the unlikely event that the <substanceUnits> are not in a molar form then more user input is required. The <initialConcentration> field represents the concentration of a substance and can only be used if a volume is assigned to a compartment. The units used in the <initialConcentration> field take the form of <substanceUnits>/<spatialUnits> (Note <initialConcentration> has only been included since the recent level 2 revision in SBML.) If the specified initial concentration is in a mole/litre amount then the molecules can be obtained by assuming one molecule is equal to 1nM [SPB01]. In the unlikely event that mole/litre concentrations are not used then more user input is required. The <initialConcentration> and <initialAmount> fields are mutually exclusive; if either of these fields is empty then it is implied that the values are unknown or are not needed for analysis. If the values are missing from a SBML file then the <initialMarking> is set to 0.

3.3.2 A Java\XML Implementation

The above mapping from SBML to a PNML model has been implemented as a prototype Java tool using JDOM [JDO04]. JDOM was chosen to facilitate the XML parsing due to its simplicity and platform independence - a JDOM/Java based tool will run on any system with a Java virtual machine. The tool takes in a well formed SBML file, and produces a standards-compliant PNML file. At present, due to the lack of a PNML extension for stochastic nets, an annotation is added to store the stochastic rate constant where appropriate. This tool was then utilised to assess the suitability of our mapping, and as a starting point for an investigation into some structural and behavioural properties of metabolic pathways, which were obtained from both the KEGG database [KGKN02] and scientific literature via the SBML website [SBM04].

3.3.3 Import and Analysis of the Glycolysis Pathways

A SBML representation of the *Saccharomyces cerevisiae* glycolysis pathway (Figure 3.6) was converted to a PNML representation of a P/T net using a prototype tool based on the mapping presented previously. This model was obtained via the SBML website [SBM04], where it is listed along with its corresponding publication [TPR$^+$00]. For the complete SBML file and resulting PNML, the interested reader is referred to [SKSW04b]. Once converted into PNML, the model was analysed using a number of Petri net tools. The *Petri net kernel* [PNK04] was used to load the PNML and convert the PNML into a format understood by the *Integrated Net Analyser* (INA) [Sta04]. INA was then utilised to calculate invariants. The model was analysed for for T-invariants and P-invariants [Mur89]. P-invariants are a set of places that retain the same total marking in any given state of the system. T-invariants are a set of transitions that when fired in a partial order, return the the state of the net to its initial marking. Invariant analysis is independent of the initial marking. The methods to calculate invariants are discussed in chapter 2. One P-invariant was discovered, relating to the tight coupling between BPG and NAD. Fifteen T-invariants relating to the reversible reactions were found. This analysis, while simple, shows how Petri net properties can be used to validate the model.

To further illustrate the presented mapping, a model of pyruvate branches in *Lactococcus lactis* [HSM+02] was also imported and converted from the SBML model repository. Abbreviations from this model are listed in Table 3.2. Again, this model was based on the glycolysis pathway. However, in this case the focus was on pyruvate branching. The aim of the original study [HSM+02] was the optimisation of the production of di-acetyl, a by-product of glycolysis, which is an important flavour component in dairy products such as butter. A SBML file of the model used in this study is available from the website [SBM04] and is graphically represented in Figure 3.9. Using the prototyped tool, this model was converted into PNML. Again, the PNML produced by the conversion tool was exported into INA [Sta04] via the Petri net kernel [PNK04]. The net contained six Place invariants, but no T-invariants. The P-invariants of this net are as follows:

$$(NADH)+(NAD)=0 \quad (3.7)$$

$$(AcCoA)+(CoA)=0 \quad (3.8)$$

$$2(Ac)+(ActoinIn)+2(AcCoAate)+(AcetoinIn)+(AcetoinOut)+(AcLac)+$$
$$(O2)+(NAD)+2(AcP)+(AcO)=0 \quad (3.9)$$

$$(ADP)+(ATP)=0 \quad (3.10)$$

$$(Ac) + (lactate) + (ActoinIn) + (pyruvate) + (AcetoinIn) + (AcetoinOut) +$$

$$(Butanediol) + (AcLac) + (EtOH) + (AcP) + (halfglucose) + (AcO) = 0 \quad (3.11)$$

$$(Ac) + (AcP) + (Pi) = 0. \quad (3.12)$$

The majority of the P-invariants here are explained by the inherent tight coupling between specific molecular species, validating the basic composition of the model. However, equations 3.10 and 3.12 appear more interesting, and may provide further insight into the pyruvate branches in the hands of the original experimentalists, perhaps providing prerequisites relating to a specific path through the model.

Abbreviation	Definition
AC	Acetate
ACCOA	Acetyle coenzyme A
AcetoinIn/AcetoinOut	Acetoin
ACLac	Acetonlactate
ACP	Acetyle phosphate
COA	Coenzyme A
ETOH	Ethanol
O	Oxygen
Pi	Inorganic phosphate
ATP	Adenosine triphosphate
ADP	Adenosine tiphosphate
NAD	nicotinamide adenine dinucleotide (oxidised)
NADH	nicotinamide adenine dinucleotide (reduced)

Table 3.2: Abbreviations from the model of glycolysis presented in [HSM$^+$02].

Finally as an exemplar of the potential of the SBML/PNML mapping every *B. subtilis* pathway in KEGG (available in SBML) was converted into PNML. The results are not shown, but demonstrate the possibility of automatically importing a large number of pathways in to the Petri net framework.

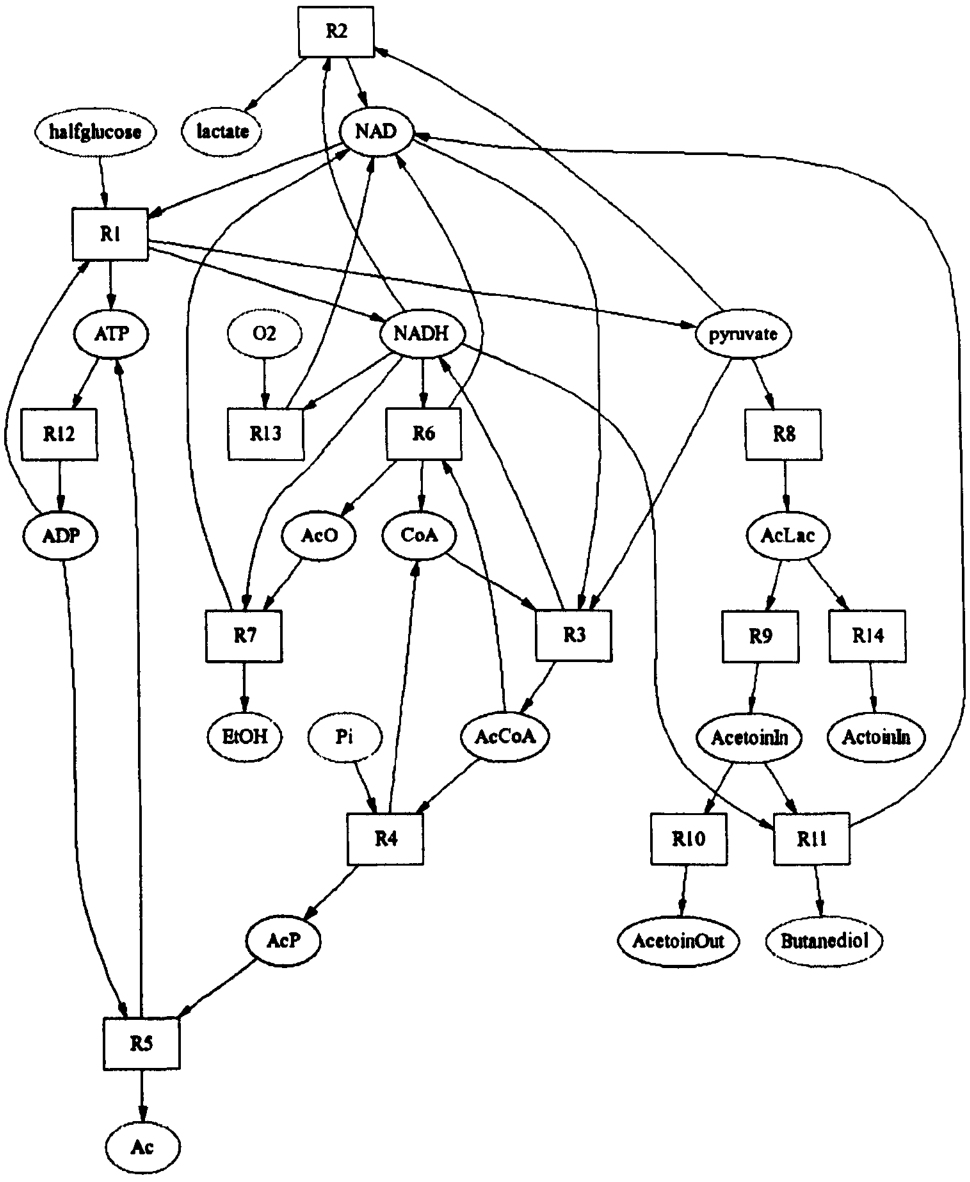


Figure 3.9: A graphical representation of the glycolysis pathway from [HSM+02]. Abbreviations are listed in Table 3.2

3.3.4 Other P/T Net Properties

Although only P and T invariants have been considered here, there are many other interesting properties that can be analysed and discovered with Petri nets that may be useful to biologists. These are now discussed including indications of how some of these Petri net properties can be related to biological systems.

- *Deadlock.* A deadlock would indicate a Petri net marking that once entered leaved no enabled transitions and hence no more transitions can fire. Presence of a deadlock would, at the holistic level, represent the death of an organism. Deadlock in smaller systems or sub models would represent a pathway no longer available to an organism. For example, the lack of a resource would lead to deadlock in that pathway and an alternative pathway would be utilised.

- *Boundedness.* Places, or a set of places, in a net can be shown to have a particular bound on token number. A particular molecule may be toxic/lethal to an organism at a certain level. Alternatively a molecule may induce further reactions at a certain threshold. Hence, boundedness analysis would be able to check if these conditions occur from an initial marking.

- *Traps.* A trap is a set of places which never lose all their tokens once marked. In biology this could represent a persistent protein or metabolite, i.e. one with no degradation pathway.

- *Siphons.* A siphon is a set of places where every transition having an output place in the siphon has an input place in the siphon. The presence of a siphon in a biological

Petri net would indicate a store of proteins or metabolites that have no way of being replenished. For example these could be metabolites that are no longer available in a particular environment.

- *Reachability*. Reachability is the systematic search of all possible state spaces. It can be determined if a particular marking is reachable from a given initial marking. Reachability is potentially very interesting to biologists. By systematically deleting places from the net (the equivalent of a biological knock out experiment) the biologist may be able to find key genes that prevent certain states from occurring. Conversely, over-expression studies may be carried out by increasing token number of places representing important molecular species.

There are undoubtedly many other biological analogies in Petri net properties that could be exploited using analysis techniques such as model checking.

3.4 Conclusions

Many systems biology studies focus on the simulation of biological systems, and the analysis of the time series data that results [ARM98, Gib00, MA97, SPB01]. These simulations should be enhanced or ideally preceded by some validation of the model in question [HK04]. In this section, the framework of Petri nets is utilised to validate models of biological networks. Two distinct methodologies are covered in this section: the analysis of Boolean representations of biological networks using safe Petri nets; and the import and analysis of biological networks utilising P/T nets. These techniques allow a researcher to validate his or her basic understanding of a model while some interesting properties may also be gleaned. Validating a model ensures that the model conforms to the researcher's assumptions, and gives increased confidence in their understating, allowing more detailed kinetic models to be produced.

The Boolean based modelling of biochemical systems, specifically genetic regulatory networks, is an important component of the study of these systems as it gives a clear indication that the basics of the system has been correctly understood. The investigation and validation of these models is interesting in its own right, clarifying the researcher's understanding of the system's dynamics. The validation of these systems is also an important component that is important for the future study of these systems with the aid of advanced stochastic or deterministic simulations.

In this section a technique presented by Steggles *et al.* [SBSW05] was utilised to represent a Boolean model of the Lac operon as a Petri net, capturing the model's behaviour utilising synchronous semantics. This approach builds on previous Boolean based techniques, such as those presented by [AKMM98]. By modelling Boolean based systems using safe Petri nets the model is immediately amenable to the advanced analysis techniques available within the Petri net framework. Importantly, this allows the efficient exploration of the state space, and the ability to cope with inconsistent or missing data.

The technique was applied to the classical Lac operon, (and later in Chapter 6 the *Bacillus subtilis* phosphate stress response). The models were described as Boolean truth tables in order to reverse engineer the Boolean model. The safe Petri net models allowed fast exploration of the model's state space via Petri net based tools such as PEP [Gra97], and the unfolding algorithms therein [Kho03]. Analysis of these models validated the original knowledge of the system. Although no original knowledge or properties were gleaned, the validation of these models demonstrates if the biological knowledge has been captured and modelled correctly. Once Boolean models have been reverse engineered, validated and refined, it is possible to construct kinetic models of biological systems from this information. The validation of Boolean models of biochemical systems can greatly facilitate the difficult process of constructing a kinetic model, with all the unknowns and missing parameters inherent in this approach [MS03].

The mapping of biochemical elements to the Petri net formalism was pioneered by Reddy [RLM96] and has been built upon in this thesis by the mapping of SBML [HF$^+$03] to PNML [KW01a] schema. This work allows a vast number of biological networks available in SBML to be analysed using Petri nets. Indeed it was possible to apply this conversion to all the metabolic networks for *B. subtilis* present in the KEGG database [KGKN02], (results not shown). This work is potentially fruitfully to both the systems biology community (for the analysis of their models) and the Petri net community (as an application domain for novel techniques). The mapping and the Java implementation was of particular interest to the PNML community, as exemplified by the publication of a paper on this work at a meeting of the PNML community [SKSW04a]. The mapping presented here is demonstrated by analysis on two models of glycolysis available from the SBML website [SBM04]. The models presented were analysed for invariants, which validated the basic structure of the model, and in the case of pyruvate branching, indicated possible interesting invariants for future investigation.

Chapter 4

Stochastic Simulation of Biological Networks, Theory and Tool Support

4.1 Introduction

Kinetic simulation is an important element in computational systems biology studies. This is exemplified by the number of kinetic simulation tools available on the SBML website [SBM04]. Kinetic simulation based studies generally require: (a) a well defined model structure; (b) initial molecular concentrations; and (c) kinetic rate constants. A simulation algorithm, via an appropriate software tool, is then utilised to generate a system's state over a given period of time. Kinetic simulations have an advantage over the other techniques discussed previously in that generation and analysis of time series data is possible. With this time series data it is possible to move from asking questions such as, "will this transition fire?" to "how many times will this transition fire on average?"; from "can we get from state A to state B?" to "how many molecules of x will there be in an average population after 10 minutes?". Depending on the simulation techniques being employed, the state of the system at infinitesimal time points can be calculated or error bounds can be placed on molecular amounts. Simulation techniques utilised for biological networks can be placed in three main categories, *Deterministic* [IBG$^+$04], *Stochastic* [ARM98] or *Hybrid* [MDNM00].

Initially, this chapter discusses the merits of the three main simulation techniques. Discussion then focuses solely on stochastic techniques. Whilst a powerful simulation technique, stochastic simulation suffers from two important drawbacks; high computational cost [EB01]; and a general lack of kinetic parameters [MS03]. To facilitate the stochastic simulation of biochemical networks, a new stochastic Petri net simulator, *NASTY* (Not Another Simulator Thank You), has been developed. NASTY is compliant with the Petri Net Markup Language (PNML) [BCvH$^+$03], and notably uses mass action kinetics [Wil06] as a default since this is fundamental to the stochastic simulation of biological networks [Gib00].

NASTY is a unique software tool, providing a number of key features:

- an efficient simulator, incorporating the Gibson-Bruck algorithm;
- default use of mass action kinetics;
- PNML compliance, and, through the mapping provided in Chapter 3, SBML compliance;
- and the distribution of jobs over remote machines.

The simulation engine is also computationally amenable and has provided the basis for all simulation related studies throughout the remainder of this thesis.

NASTY addresses the computational cost of carrying out stochastic simulations by utilising an efficient algorithm, based on the Gibson-Bruck algorithm [GB00] combined with stochastic Petri nets [Mar89]. NASTY further reduces the real time computational costs of multiple simulations by distributing the processing over multiple processors to ensure effective use is made of all available processing power.

Whilst NASTY provides a powerful tool for the stochastic simulation of biochemical networks, there was still another technical problem that needed to be addressed. To allow full exploitation of time series results arising from stochastic simulations, these results must be stored, transferred and be made amenable for analysis. To assist in this, a standardised results format was devised. *Time Series Markup Language* (TSML), is a novel results interchange format for both stochastic and deterministic time series data. TSML provides a schema allowing results to be stored in an efficient, loss-less manner. TSML does this by tracking the changes rather than storing all the data from the simulations. By doing this, all the original information can be constructed and statistically analysed. Both NASTY and TSML are taken forward for use beyond this chapter, and prove central to the work in future chapters.

4.2 Simulation Techniques

There are three broad categories of simulation techniques available for use in systems biology: *deterministic* [IBG$^+$04]; *stochastic* [ARM98]; and *hybrid* [MDNM00]. These simulation techniques are described in chapter 2 Deterministic and stochastic simulations have a number of advantages and disadvantages, while hybrid techniques aim to combine the best of both approaches (some disadvantages of the original techniques still remain). Hybrid simulation techniques are not considered further here as they do not have the theoretical justifications and history of the deterministic and stochastic techniques.

Due to the accessibility of tools and ease of computation, deterministic techniques have proved popular in systems biology studies. This is exemplified by the large number of curated models accessible via the SBML website [SBM04]. As with all simulation techniques, deterministic modelling relies on a number of assumptions, namely:

- that concentrations are sufficiently large so there can be a mid point between two concentrations;
- that the system behaves in a fully deterministic way;
- and changes over time are continuous.

Deterministic algorithms allow fast and accurate simulations of these systems. While deterministic algorithms and techniques have been used frequently, there have been fewer examples of detailed stochastic studies. However, this area is growing with some detailed discussions [Gib00] and example models [ARM98]. In systems with small molecular amounts, assumptions justifying deterministic simulation break down. In systems such as these stochastic simulation may be utilised. Stochastic simulations are based around Monte-Carlo based algorithms [Gil76, Mol82]. Randomness is inherent in biological systems [MS95], and stochastic effects could well account for the non-genetic, non-environmental differences observed in biological populations. The main assumptions involved in stochastic modelling are as follows:

- the Rates are well know;
- the initial amounts of molecules are well known;

- and the reagents are well mixed.

There are many arguments for the relative benefits of stochastic and deterministic modelling, and a detailed discussion of these is presented in Chapter 2. In the remainder of the thesis, stochastic simulations are focused on, as they:

- capture fine grained behaviours;
- allow different outcomes in the same model, and so non genetic variations in a population can be discovered;
- have a theoretical justification for small biochemical systems [Gil76];
- and have theoretical assumptions that are always valid, when deterministic assumptions are valid [Gil77].

Stochastic Petri nets [Mol82, Mar89] are used in the modelling framework for the NASTY simulator.

4.3 The NASTY Simulator

In order to assess the applicability of stochastic Petri nets to systems biology, a Java tool has been developed, Not Another Simulator Thank You (NASTY). While a number of stochastic Petri net simulators already exist [PNW04], none matched all of the necessary requirements, namely:

- complete computational access;

- PNML import and export (and via the use of the mapping in Chapter 3, SBML compliance);
- integrated use of mass action kinetics;
- use of a cluster of machines to carry out multiple simulations.

NASTY was designed with three main elements in mind: a core stochastic Petri net simulation engine based on the Gibson-Bruck algorithm [GB00]; a user friendly GUI interface for the construction of models; and a distributed job scheduling protocol to allow simulations to be carried out on multiple machines. The architecture of NASTY is shown in Figure 4.1.

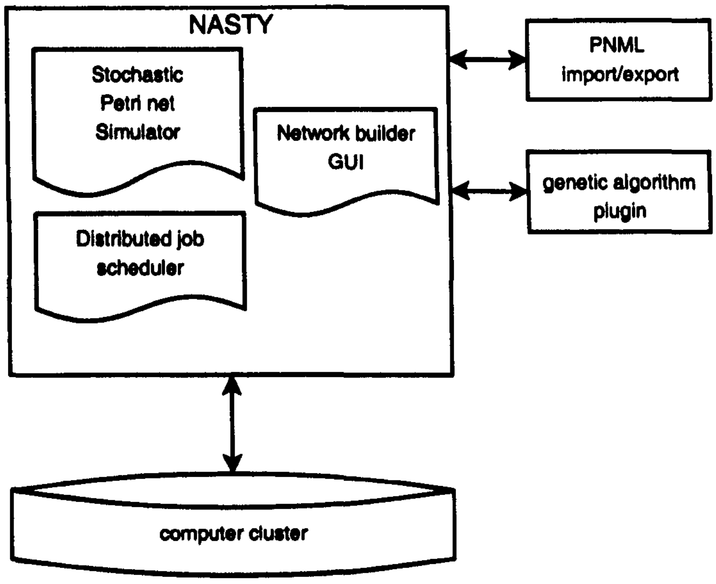


Figure 4.1: A conceptual view of the NASTY simulator

The NASTY tool was developed to provide a suitable environment for the stochastic simulation of biological networks and to this end uses mass action kinetics, which are key to stochastic modelling [Gil76] as a default.

Mass action kinetics assume that the hazard for two molecular species colliding and reacting in a constant volume at a constant temperature is constant [Wil06]. Building on this assumption the collision hazard h_i and the stochastic rate constant c_i can be calculated for each reaction i dependant on the number of species involved. NASTY currently supports only *zeroth*, *first* and *second* order reactions. These are implemented as follows;

- **Zeroth order** This represents the constant influx of some chemical species. As there is no input to the reaction the hazard is simply the reaction constant
- **First order** Here the species X undergoes a reaction. The c_i represents the hazard of a particular molecule, however when there are x_j molecules of X_j giving a combined hazard of $c_i x_j$
- **Second order** Again c_i represents the hazard of a pair of molecules X_j X_k reacting. However there are $x_j x_k$ different pairs of molecules which may react, hence the combined hazard is $c_i x_j x_k$. When two of the same molecules are involved in a reaction the combined hazard becomes $c_i(x_j(x_j - 1)/2$.

The hazard of higher order reactions can be calculated similarly. As mass action kinetics is crucial to biological modelling, the methods involved in firing a transition took into account the hazards defined above as a default.

To enable the cross platform usage of NASTY (essential for the widest user base and utilisation of compute resources) the simulator was written in pure Java [SM07]. Stochastic Petri nets [Mar89] provide an exact method for the simulation of stochastic systems, allowing the exploration of the underlying Markov chain [Mar89] (see section 2.4 for more details). However, traditional implementations of stochastic Petri nets are not as efficient as the algorithm presented by Gibson and Bruck [GB00]. The main advantages of the Gibson-Bruck algorithm come from the use of an indexed priority queue and a dependability graph. The NASTY tool aimed to implement the Gibson-Bruck algorithm by using a Petri net data model In the implementation of NASTY an object oriented view of a Petri net was used as a programmatic data model. Here objects representing places, transitions and arcs were encoded. The encoded data objects allowed the places and transitions to be connected via links to other places/transitions as appropriate. These links were the equivalent of Petri net arcs. With this data structure in place it was possible to utilise the underlying Petri net as the as the equivalent of the dependability graph of the Gibson-Bruck algorithm [GB00]. This allowed simple method calls to track the transitions that needed re-sampling after a transition firing via the change in the connecting places. Once the re-sampling has been carried out, the transitions that are currently enabled must then be monitored in an indexed priority queue, the second major component of the Gibson-Bruck algorithm. In NASTY's implementation the priority queue was implemented by the addition of certain house keeping methods to a linked list. This involves the correct placement of an enabled transition in the queue. Once in the queue the transition either moves to the top as other transitions fire, or is removed. When the transition is re-sampled, it is first removed from the list and then added again. Finally method calls were implemented that returned the transition of at the top of the queue with its associated delay. This delay is then used to monitor the progression of time in simulations. With these methods and data structures in place the underlying simulation algorithm in NASTY effectively becomes equivalent to the Gibson-

Bruck algorithm, with the Petri net fulfilling the role of the dependability graph allowing the efficient update of propensity functions [GB00].

With user interaction a key component in model building and refinement a convenient graphical user interface (GUI) was necessary. NASTY provides a Java swing GUI [SM07] which allows the user to build models by hand. Places, transitions and arcs can be inserted, moved, modified and deleted. Having a GUI allows the modeller to visually inspect their models. A visualisation of the simulation is also available, giving the modeller visual feedback on how their simulation is progressing. The combination of these graphical features provides an attractive tool for Petri net modelling.

As well as GUI inputs an important feature of a simulation tool is the import and export of standards compliant files. NASTY is compliant with the Petri Net Markup Language (PNML) [BCvH+03], allowing users to import or export models from/to the wide range of existing Petri net tools [PNW04]. File standards are also important in the systems biology community, exemplified by the Systems Biology Markup Language (SBML) [HF+03] which is becoming the *lingua franca* of systems biologists. To facilitate the use of NASTY as a biochemical simulation environment, the Java tool for interchanging PNML and SBML models described in Chapter 3 can be utilised [SKSW04a]. Hence, NASTY can be seen as having real practical applications for the biological modelling community.

Currently there are no standardised tests for a stochastic simulator. In order to evaluate the correctness of the NASTY simulator, a number of small well-studied models were investigated. A model of the Lotka-Volterra predator-prey system from [Wil06] was used to ensure that NASTY dealt correctly with fluctuating stochastic systems. The Lotka-Volterra

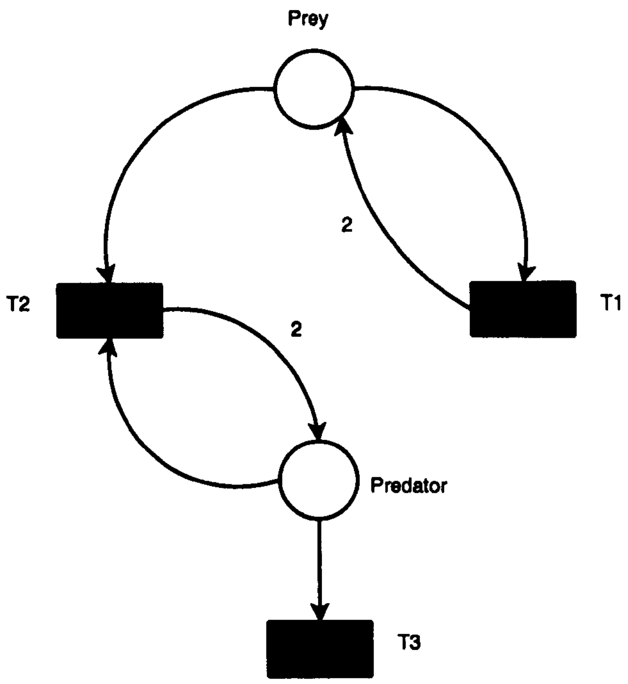


Figure 4.2: A Petri net model of the predator prey system presented in [Wil06]. Model parameters: T1=1; T2=0.05; T3=06; initial predator number 200; initial prey number = 50. Time is defined in arbitrary units.

system presents a simple predator prey relationship. The predators can eat the prey, the prey can reproduce and the predator can die. The Petri net representation of this is shown in Figure 4.2. The results from this model are presented in Figure 4.3, showing that NASTY correctly deals with fluctuating levels of tokens. Secondly, a subset of the Arkin model [ARM98] presented by Gibson and Bruck, [GB01] was considered to validate the correctness of the NASTY simulation tool. This model subset depicts translation and transcription events in *E. coli*. The results from simulation this model in NASTY are presented alongside the data from Gibson and Bruck in Figure 4.4. The output from NASTY almost exactly matches the data from Gibson and Bruck. Finally it is noted that the successful reproduction of a model of the *E. coli* stress response [SPB01] in Chapter 5 was the final validation,

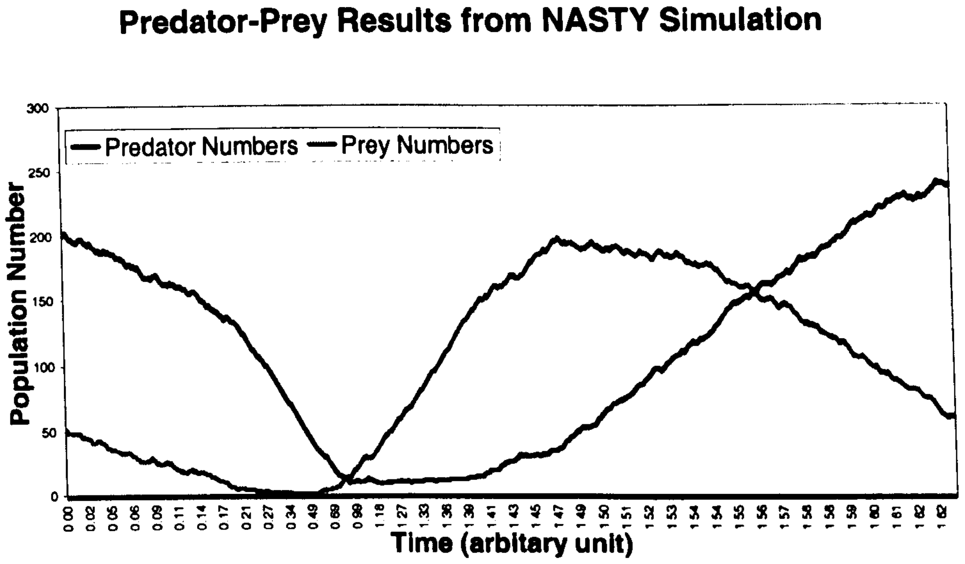


Figure 4.3: Results from the Petri net model of the predator prey system shown in Figure 4.2. Time is defined in arbitrary units.

giving great confidence in the tool.

As discussed previously, performing multiple simulations can require a prohibitively large amount of computation time, with 100 minutes of a 100 reaction system taking up to a days computation [EB01]. However, since each individual simulation is an independent job the task is straightforward to parallelise. NASTY exploits this fact and works by distributing jobs to a large number of machines to make performing multiple runs computationally feasible. NASTY can employ two methods of parallelising jobs:

- By using a number of NASTY simulation engines running as servers;
- By using the jobs scheduling tool, Condor [TTL05]

NASTY can run in two modes, server or client. The client mode is essentially the stand alone NASTY GUI. The server mode is a stripped down version, containing the core simulation tool and some network "glue" using Java sockets [McL01] through which Java

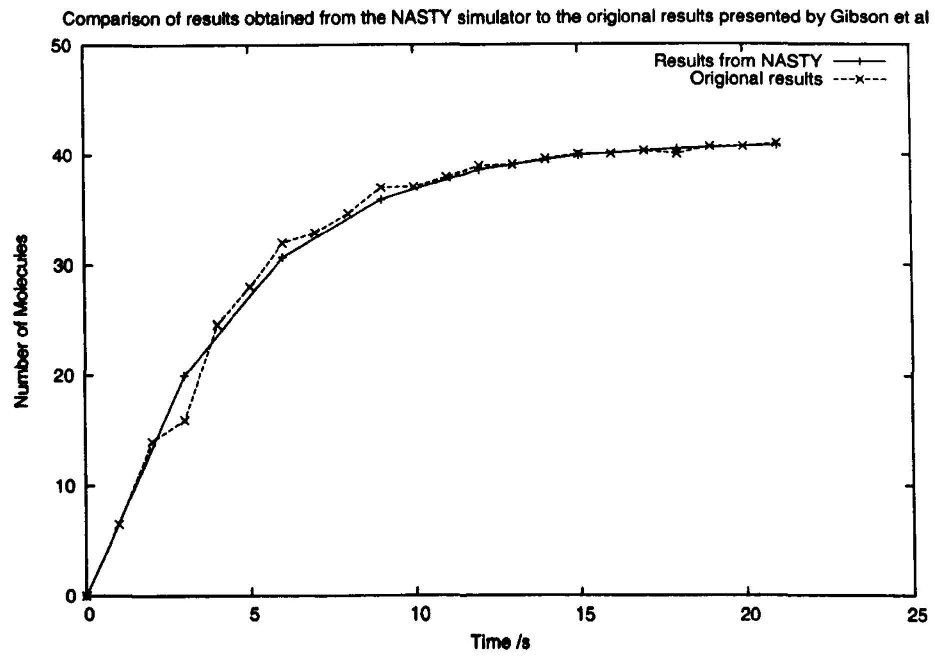


Figure 4.4: Results from the NASTY representation of the transcription translation model presented in [GB01].

objects representing models, simulation parameters and results are passed. The number of jobs and their parameters (e.g., simulation time) is created by the NASTY client. Jobs are then sent out to NASTY servers on a first in, first out basis. This requires the NASTY servers to be started on remote machines and the NASTY client to have of a list of server IP addresses. The start-up of clients is achieved through build script that starts NASTY servers over a cluster and records the successfully started servers IP addresses. As the simulations proceed each NASTY server initially receives a model and a set of simulation parameters. The NASTY client maintains a list of the availability of known servers. As the batch job progresses the NASTY client sends jobs and parameters to available servers. When processing a job a server is marked as busy. Once a job is completed the server is deemed available and may have more jobs sent to it. Each job is simulated utilising the core simulation engine of NASTY, and the results sent back to the central server and collated. This method of network distribution is ideally suited to exploratory studies where feedback and processing is required as the jobs are being carried out. Indeed, this approach is utilised in the genetic algorithm studies in the next chapter.

The core simulation engine of NASTY can also be applied via the use the job scheduling system condor [TTL05]. Condor aims to support high-throughput computing over a large collection of distributed computing resources. For example, at a University there may be times when clusters are unused (for example at night or at the weekend). Condor aims to utilise this downtime across an organisation and give users access to vast computing resources. At the university of Newcastle there is a condor pool of over two thousand computers. This resource can be utilised effectively by NASTY. NASTY, packaged as a Java jar file, a PNML file of the model and a text file with simulation parameters is first sent to condor. Condor then distributes these jobs and calls the functions that allow NASTY to carry out the jobs on remote machines. As NASTY is written in pure Java, job submission is allowed on Linux, Windows, and Mac machines running condor. While submission to condor gives vast computing resources, there is no simple way to allow feedback to a core server (as in the previous model). As a result, NASTY simulations via condor are ideally suited to batch simulation jobs where the job requirements are known in full before submission. Indeed, this technique is utilised in the sensitivity analysis in the next chapter.

4.4 Time Series Markup Language

In order to model stochastic systems, the time series output from the simulations must be available for analysis. This analysis can simply be the presentation of a single simulation run as a graph, or statistical analysis of a number of multiple runs.

Currently all simulation tools produce a "one time display" of the results of a simulation for specific species specified by the user. However, stochastic simulation of biological models is computationally intensive. For example, it has been estimated that simulating a system with 100 reactions for 100 minutes would take 1 day on a personal computer [EB01]. Whilst this is a pessimistic estimate due to ever-increasing processing power, it illustrates the problem of simulating a whole cell model with 10^{15} reactions.

It would be advantageous for modellers to be able to run their simulations once, and have the data in a readily amenable form for analysis in future. This availability may range from having the results on a personal computer, through to storing results of a large number of multiple simulations on a central server. There are a number of features that a results standard should possess:

- **completeness**-when investigating stochastic based models, subtle variations may lead to interesting conclusions. The format should effectively be "loss-less", i.e. all data should be maintained;

- **adequate metadata**-not only are the results of the data important, but also details of how they were obtained. Results should include information on the precise model and simulator utilised, along with the author and date.

- **average results**-often, a researcher is interested in the average of a number of simulation runs, possibly including confidence bars. A results format should allow the storage of this data, either alongside complete individual runs or as a average only format.

- **computational amenability**-the results format should be amenable to computation. The implementation of choice for standardisation formats at present is XML [W3C04]. XML clearly standardises the data format across multiple operating systems and facilitates implementation with object oriented databases. XML also has a number of libraries available to programming languages to facilitate implementation [W3C04];

- **human readability**-while XML standards are designed to be machine readable, the human eye is still crucial for implementation and analysis, and good XML standards should be human readable in a compliant web browser.

There are a number of common tools available for the visualisation of time series. Most prominent are common office spreadsheet applications such as Open-Office or Microsoft Excel. These tools are adequate for the representation of time series data with a constant time step, for example data in a form such as Table 4.1

time/seconds	SpeciesA	SpeciesB
10	10	6
20	11	6
30	12	6
40	13	7
50	14	7

Table 4.1: Time series data with a discrete, uniform time step

However, stochastic simulations results do not have uniform time steps, due to the random sampling of the propensity functions. Stochastic systems evolve through the occurrence of reactions that happen at non uniform time intervals. Another important quality of these simulations is that the result of a single reaction is sensitive to a change in a small number of molecular species in the system. Monitoring every species for every reaction is clearly inefficient. A more efficient results format would track the changes in the system, noting the number of molecules a place has at a particular reaction time. For example:

Species A { (10, 1.07232), (11, 2.5342), (12, 7.342) }

Species B { (6, 3.6548) }

Here species *A* is involved in three reactions while species *B* is only involved in one. Therefore, recording the time of each individual reaction and the resulting change in the number of molecules is highly efficient, and would offer a loss-less form of data storage.

This putative Time Series standard was implemented in XML, as a proof of concept, in collaboration with a project student. The standard is given the name Time Series Markup Language (TSML). The features of this XML standard are represented in Figure 4.5 and can be summarised as follows. The TSML standard is split into two main elements, `Results` and `Metadata`. `Metadata` has a number of components to represent the full information on the model, which is crucial, especially when the differences between simulators and models and the effect this may have on the results is considered. The sub-elements of `Metadata` are:

- **Title**-the title for the results, preferably a meaningful name for fast look up;
- **Model-sources**-a URL of the model utilised for this simulation. It is vital to state

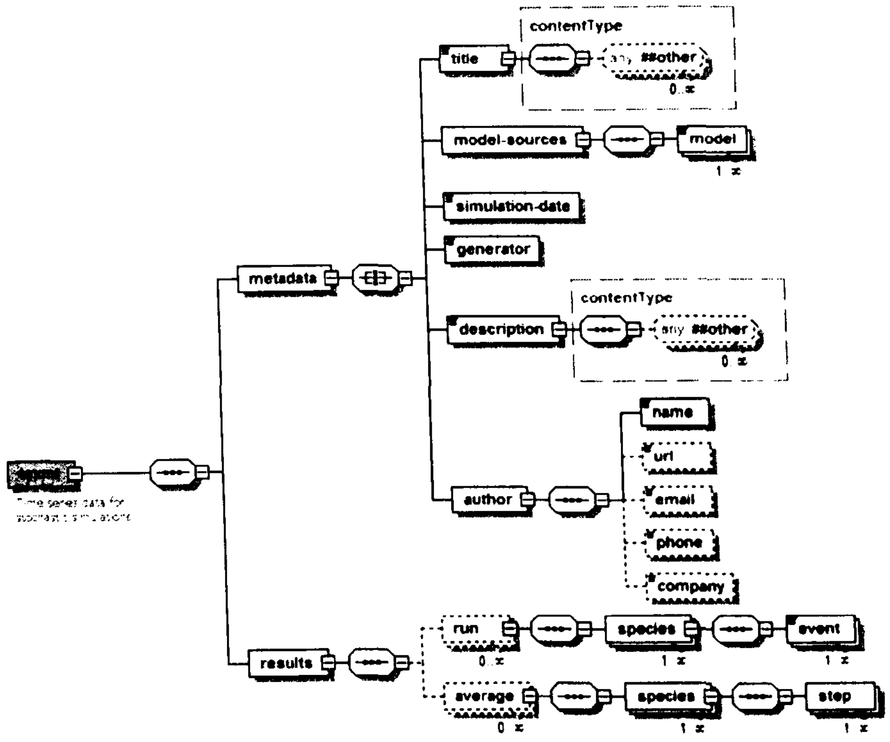


Figure 4.5: The TSML schema

the model used. For example, if a model improves then the results may become redundant;

- **Simulation-date**-to track the date that the simulation was run;
- **Generator**-it is important to know which simulator these results came from. There are many stochastic tools available, but only preliminary work on their correctness. As a result, the quality of data from these simulators may be low;
- **Description**-a field that allows the researcher to specify the aims and objectives of their experiments;
- **Author**-the author of the study, including name, email and institution.

The main results are stored in the `Results` element. The results element has two optional sub-elements: `run` and `average`, which represent and individual run and the average of a number of runs respectively. A TSML file may have any number of `run` objects, and it may or may not have an `average` element. TSML file may contain have just an `average` element. However, this may not hold enough information for future work on the results. A `run` element contains a number of `species` elements. These species also contain a number of `event` elements, containing the time of a reaction and the resulting molecular amount after that reaction has occurred. An `average` element also contains a number of species. The species contain `step` elements that have a `time` unit and an `average number` unit. This represents the average number of the molecular species at a given time point.

This standard has greatly assisted the work in this thesis on stochastic simulations. A tool was also developed by a student (J.Marcelino), sponsored by the Wellcome trust, that allowed an interface between compliant TSXML files and the plotting tool gnuplot. This tool allows the automatic plotting of a simulation run, via a Java interface. It is hoped that TSXML will become the standard for time series data. A technical report is in progress describing the format in detail along with the Java interface to the plotting facilities.

4.5 Summary

This chapter is concerned with the tools that support the stochastic simulation of Petri net based models of biological systems. The thorough theoretical rigour provided by Gillespie [Gil77, Gil01] allows researchers to pursue the stochastic simulation of biochemical systems. The field of Petri nets, specifically Stochastic Petri nets (SPNs) [Mol82, Mar89] allows an appropriate formalism by which these simulations can be carried out. SPN's have been utilised in a number of studies into biochemical systems [GP98, TL02, ML97, SPB01]. All these studies have been carried out with the Mobius tool [CCD$^+$01], or a prequel to this. This chapter demonstrates how the Gibson-Bruck algorithm [GB01] and SPNs can be amalgamated to produced an efficient platform, both efficient and exact, within the Petri net framework for the simulation of biochemical systems. A Java tool, NASTY was developed to demonstrate this SPN framework. The tool has a number of further advantages over current simulators for the work in this thesis: default use of mass action kinetics, complete computational access; networked simulation over multiple CPU's; and the import and export of PNML. The tool has been tested on a number of small models, giving confidence in the tools performance. NASTY allows simulations to be carried out over multiple CPU's, either by the distribution and allocation of jobs within NASTY, or by the packaging and simulation of jobs via condor. The distribution of jobs is highly important to the investigation of stochastic networks via the use of multiple simulations.

Finally, in this chapter a format has been devised for the storage and interchange of the results of stochastic simulations. TSML (Time Series Markup Language) was developed in an attempt to mitigate against the need to carry out computationally expensive simulations every time simulation results were required. TSML allows the storage of individual runs of stochastic simulations. This is argued to be important as stochastic effects can show some non-genetic, non phylogenetic differences between an individuals of isogenic population [MA97].

The work in this chapter introduced valuable tool support for the stochastic simulation of biochemical networks. NASTY, with its efficient simulation engine, inbuilt mass action kinetics and network distributed simulation capacity, has the potential to be of real benefit to the community. NASTY is available as a download on request from the author. TSML is under current development and is to be used and further developed in the author's ongoing research.

Chapter 5

Parameter Estimation and Sensitivity Analysis

5.1 Introduction

In the previous chapters stochastic Petri net techniques were discussed and developed as a means of modelling and simulating biological networks. This work was motivated by the fact that stochastic Petri net simulation techniques have been shown to capture the fine grained behaviour and randomness of outcome of biological networks not captured by deterministic techniques [MA97, IBG$^+$04]. Furthermore, stochastic simulation methods, such as stochastic Petri nets, have been argued to be more suitable to small (single-cell) [GB00, Gil77] biochemical systems, and are at least as suitable as other techniques [Gil76]. Thus, stochastic simulation techniques are becoming an increasingly important aspect of systems biology studies [ARM98, MA97].

While stochastic simulations provide a powerful tool for systems biology, current efforts in the stochastic simulation of biological networks are hampered by two main problems:

1. There is a large computational cost to the stochastic simulation of these networks. A recent review suggested that an average personal computer would take a whole day to simulate 100 minutes of a 100 reaction system [EB01]. This problem is exacerbated since multiple repetitions of simulations are usually required.

2. There is a lack of quantitative data relating to the molecular concentrations and kinetic parameters that are essential to the successful simulation of biological networks [MS03].

The first of these problems was considered in Chapter 4 where a new stochastic simulation tool NASTY for stochastic Petri nets was developed, allowing the computational cost of simulations to be mitigated against via a cluster of computers. In this chapter, the problem of incomplete kinetic parameter data for stochastic Petri net models is addressed by developing parameter estimation techniques [MK98, MMB03] based on a well known optimisation method, genetic algorithms [Hol75]. To increase the understanding of a models dynamics in an aim to improve parameter estimation, sensitivity analysis is applied [GCPD05]. Knowledge gained from sensitivity analysis should be utilised in order to improve parameter estimation methods, both by giving search priority to the most sensitive rates, and focusing on a smaller range of values for others. Both automatic parameterisation and sensitivity analysis are computationally costly [GB01], a challenge that has limited applications of such techniques to stochastic models of biological systems.

5.2 Parameter Estimation

The ideal scenario for a systems biology study is that the model structure is composed, the kinetic parameters and initial conditions are well defined, and then the model is dynamically simulated [GB01]. However, this is usually not possible due to the problem of unknown kinetic parameters [GP98, GB01, MK98, MMB03], which is sometimes seen as an "insurmountable" problem for quantitative kinetic modelling [MS03]. Until there are accurate parameters for all the kinetic reactions in a biological system the reverse of the idealised modelling procedure must be carried out, that is using observed data from laboratory studies to infer missing parameters [MK98, MMB03, GB01]. The kinetic parameters, once estimated, can hopefully be refined via the iterative systems biology process of laboratory studies and quantitative simulations [IWK+04], accurately measuring key parameters and honing the model.

Parameter inference is far from a trivial exercise. Typically, models of biological systems have many interacting variables and an infinitely large potential parameter set of floating point values, so state by state exploration is prohibitive [MK98]. Due to the potentially infinite set of possible kinetic parameter values to be explored, parameter estimation must intelligently circumvent this problem, utilising other, more advanced techniques [MMB03]. There are a number of ways of intelligently exploring the state space of a system with many missing parameters. For example, Linear programming is commonly utilised to explore systems with multiple missing parameters [MMB03] which is lacking in these stochastic systems. However these techniques cannot be applied to biochemical networks due to their requirement of objective function in terms of the adjustable parameters [MK98]. There are also direct search methods such as the Newton method [MMB03], but these techniques are local minimiser, and are not suitable for biological systems with many local minima, as the global minima may be missed [MK98].

A range of parameter estimation methods are suitable for kinetic biochemical models, with many having been practically applied [MK98, MMB03]. First, there are a number of advanced mathematical methods for inferring the value of missing parameters, and indeed these have been investigated in the context of stochastic models of biochemical networks. Gibson reversed his simulation algorithm [GB00], and with knowledge of the systems time trajectories a missing parameter was inferred [Gib00]. This method, while potentially important during the fine tuning of a biological model, is not well suited when there are multiple missing parameters. Bayesian inference has also been applied to the problem of missing parameters [GW05]. Here the authors utilised a Markov Chain Monte Carlo Method to statistically infer a small subset of unknown parameters of a small, artificial model of a biochemical system. The artificial model under investigation here was stochastic, however the authors utilised a stochastic differential equation approach, suggesting that while diffusion approximation is inappropriate for single cell model, it is satisfactory for use in a Bayesian inference algorithm of such models [GW05] . This study found suitable parameters for the small system under study, however the technique required very detailed knowledge of the trajectory of all the molecules in the system, which is not typically available [EB01].

A common approach utilised for the search of missing parameters is that of probabilistic optimisation algorithms [MK98]. These techniques are designed to find a global minima over a search space with many local minima, and are ideally suited to the search for missing global parameters [Hol92, KGjV83]. There are numerous probabilistic optimisation algorithms available in the literature, such as simulated annealing [KGjV83], tabu searches [ZL02], and genetic algorithms [Hol75]. Probabilistic optimisation algorithms have been applied to the investigation of deterministic models of biochemical networks [MK98, MMB03], however because of the computational costs involved, they have not been applied to stochastic biochemical networks [GB01].

The search for missing parameters in biochemical systems suffer from the curse of dimensionality. Here a set of parameters can combine combinatorially to produce and infinite search space. With stochastic reaction constants having a floating point precision, and states of the reactants being unbounded this problem is compounded. Thus the automatic parameterisation of a kinetic model of a biochemical system is thought to be an NP-complete problem [MMB03]. Such NP-complete optimisation problems have no known efficient algorithms for finding exact solutions, due to the exponential state-spaces involved [GJ79]. Probabilistic optimisation algorithms aim to locate the optima of a function which represents the requirements of the performance of the final system [KGjV83]. This involves creating an approximate function that describes the problem and finding a set of parameters that maximise (or minimise) this function. In these NP complete problems there may be many local minima, however probabilistic optimisation algorithms are concerned with optimising a system to find an approximation to the global minima (or maxima). Probabilistic optimisation algorithms do not require an explicit function (the behaviour of the system being completely characterised by a single equation). Instead they require a cost function to be evaluated only at certain points in the state space, thus the results of kinetic simulations form a suitable candidate for many probabilistic optimisation algorithms [MMB03].

Parameter estimation techniques have proved useful in a range of diverse fields, for example in finance [DM01], logistics [GJ79] and the creation of putative genetic networks [SHSW04] and they have been studied in some detail in the field of kinetic biochemical models [MK98]. However, these studies are restricted to deterministic systems. For example, in a recent review [MK98] a comparison of random search, simulated annealing, genetic algorithm and evolutionary programing methods suggested that a number of different optimisation techniques should be utilised, maximising the possibility of finding suitable parameters. A follow up review [MMB03] analysed a large system with thirty-six parameters and utilised a number of methods. The review of three optimisation methods suggested that an genetic evolution based algorithm performed the most effectively, and was the only method able to solve their system [MMB03]. The global optimisation of deterministic systems has, to some extent, been well investigated, as exemplified by the previously mentioned review papers [MMB03, MK98]. However, there does not appear to have been much progress in this area for stochastic techniques, and in fact it has been noted as an important area for future work [GB01]. Probabilistic optimisation algorithms, by their nature, require the optimisation function to be evaluated a large number of times. In deterministic systems this requires a full simulation of the model, and indeed this is the case for stochastic systems. However stochastic simulations are far more computationally demanding than deterministic simulations and have thus proved problematic in the past [GB01].

5.3 Developing a Parameter Estimation Algorithm

The NASTY simulator was utilised to mitigate against the high real-time cost of performing the necessary stochastic simulations [Gib00] required by a probabilistic optimisation algorithm. As described previously, NASTY allows jobs to be sent out over a cluster of machines, effectively parallelising the computation. With a number of probabilistic search algorithms available, a decision had to be made on the most suitable approach for this problem. The usual choices of evolutionary computation [Hol75], simulated annealing [KGjV83] and tabu search [ZL02] were considered. Various probabilistic probabilistic search algorithms have similar performance when applied to deterministic simulations [MK98, MMB03]. For the problem of stochastic network parameterisation it was decided to utilise a *genetic algorithm* [Hol75], for two main reasons. First, review papers suggested that evolutionary computation was slightly more suitable for problems like these [MMB03], if in fact any technique was preferable [MK98]. Secondly, the computational cost could, at present, only be alleviated by using parallel computation (that is separating a problem into distinct, self-contained computational units). To harness a large number of machines, requires the ability to send out individual, well contained simulation jobs. Simulated annealing, for example, carries out a large number of simulations in a sequential manner. This would obviously not benefit from the cluster compute facilities available via NASTY. Genetic algorithms however, rely on the evaluation of a population of individuals [Hol75]. This allows NASTY to send a population of solutions to be simulated in parallel over a cluster, allowing each machine to run an individual, self contained job and thus effectively making use of the processing resource.

A genetic algorithm, based on the ideas presented in [Hol75] was developed, using NASTY as the job scheduler and simulation engine. The genetic algorithm approach is based on a simplified interpretation of Darwinian evolution [Hol75]. Here a population of individuals is allowed to evolve [Hol75], with fit individuals being more likely to survive to the next generation. Over time, this creates a population of highly fit individuals. In the genetic algorithm approach, individuals are represented by a single "chromosome" that encodes a possible solution to the optimisation problem. Each chromosome is then evaluated for correctness against a *fitness function* which determines how well the solution encoded in the chromosome relates to a number of desirable qualities. Thus the fitness function is a key component of the genetic algorithm and needs to be carefully constructed [Hol75]. The fitness of the individual relative to the population is equivalent to the likelihood that the individual genes progress to the next generation. The idea is to allow a population of solutions to evolve using techniques analogous to those found in real organisms, such as "crossovers" where the chromosomes mix, "mutations", where there is a random change on a gene in the chromosome and "cloning", where an individual proceeds unchanged to the next generation [Hol75]. The evolution of fitter individuals is favoured, but not guaranteed. The population remains diverse due to crossover and mutation events, which prevent the population tending to a local minima.

In the resulting implementation of the genetic algorithm, each possible solution to a network's parameters is represented by a single "chromosome", where each kinetic parameter is representative of a "gene" [Hol75]. Each chromosome contains one copy of every "gene" (reaction rate parameter). The genes are ordered relative to the reaction number.

These chromosomes are simply modelled as vectors of floating point numbers (genes). The fitness of each individual solution (chromosome) is then calculated by the fitness function. To calculate the fitness of an individual its parameters are utilised to simulate a realisation of the system. The results obtained from the given solution are then compared those obtained from the "gold-standard" parameters. The fitness function uses a combination of Pearson's correlation coefficient and Euclidean distance and is listed below.

$$\left(\frac{\sum_{i=0}^{n}(100 - \frac{result_i \times 100}{gold_i})}{n \times 100} + r\right)/2 \tag{5.1}$$

Here there are n time points that are being measured. For each time point i there is both a result from the obtained parameters, $result_i$, and a result from the gold standard parameters, $gold_i$. The percentage distance from the gold standard is calculated for each time point, and summed to give a value between 0-1. Finally the Pearson correlation coefficient r is calculated and added. The total is then divided by 2 to give a value between 0-1. This cost function forms a central part of the resulting genetic algorithm, which is presented below in pseudo code form (Algorithm 5).

The algorithm starts with the initialisation phase in which an initial population, P_0, of individuals is created by randomly creating vectors of rates, where each rate, r, is in the range $0.0 < r < 1.0$. The population is then simulated to allow the fitness of each individual to be assessed. This stage is typically intractable for stochastic simulations due to the large compute effort [EB01]. To mitigate against the real time simulation costs, the NASTY simulation engine, described previously in Chapter 4, is utilised. Each individual is simulated a number of times to obtain an average of its components time trajectories.

Algorithm 5 The Genetic Algorithm

```
1: Initialise Population $P_0$
2: for $g = 0$ to $MAX$ do
3:    for Solutions $s \in P_g$ do
4:       Simulate $s$ to calculate its fitness
5:    end for
6:    Create new empty population $P_{g+1}$
7:    while $Size(P_{g+1}) < (Size(P_g) - CLONES)$ do
8:       Select $s1$ and $s2$ from $P_g$ using fitness values
9:       Crossover $s1$ and $s2$ to produce $s3$ and $s4$
10:      Add $s3$ and $s4$ to $P_{g+1}$
11:   end while
12:   while $Size(P_{g+1}) < Size(P_g)$ do
13:      Select $s$ from $P_g$ using fitness values
14:      Insert $s$ into $P_{g+1}$
15:   end while
16:   for Solutions $s \in P_{g+1}$ do
17:      if $Random() < MUTES$ then
18:         Mutate $s$ within $P_{g+1}$
19:      end if
20:   end for
21: end for
```

This is done efficiently by NASTY by farming out the simulation tasks to the server pool. The resulting average time trajectory is then compared to a gold standard to obtain a fitness score for the individual. The fitness of the whole population is then calculated by summing the individual fitness scores.

The next step is to begin the selection process for the next population. In our approach a random roulette wheel based technique [Gol89] is used to probabilistically select individuals which are then subjected to one of three fates [Gol89]: *crossover*; *cloning*; and *culling*. During crossover two individuals are selected and these then "breed" to produce two new children by randomly selecting genes from the two parents (see Figure 5.1.a). These children then pass into the new population. Individuals may also be selected to progress unchanged to the next population and we refer to this as cloning (see Figure 5.1.b). Any individual not selected for cloning or crossover has effectively been culled and will not appear in the new population. The number of individuals cloned is governed by the constant *CLONES* in the algorithm above (10% in the implementation used). Any individual that is not cloned is then subject to a crossover event. Note that using this approach the fittest individual may not survive and conversely, the least fit may. This is an important point since it helps prevent the population getting stuck in a local minima.

Once the makeup of the next generation is decided the mutation phase begins. Here individuals are selected randomly to be subjected to a single random gene mutation, as shown in Figure 5.1.c. This equates to a single parameter taking on a new random value between 0 and 1. The number of mutations applied is controlled by the threshold constant *MUTES* (5% in this implementation) which sets the probability of performing a mutation. These mutations introduce new genes into the population, giving the potential for more varied solutions to be considered. After the mutation phase is completed a new population emerges. The whole process above is then repeatedly applied until a pre-defined number *MAX* (50 in this implementation) of populations have been generated.

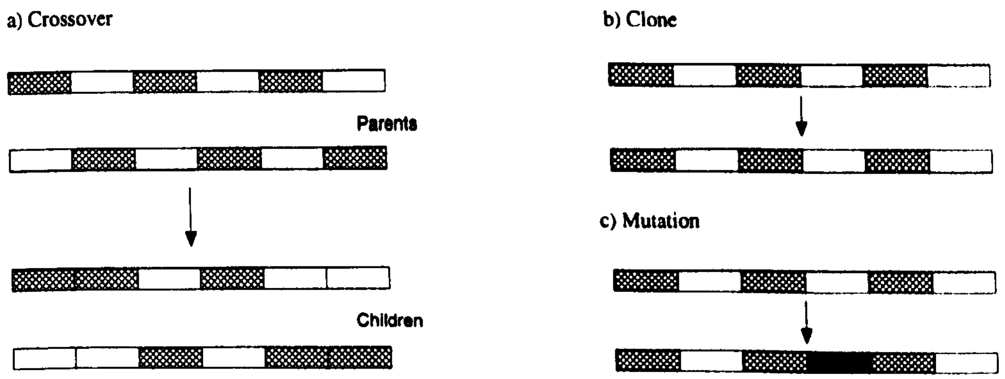


Figure 5.1: Possible fates of solutions during the evolution process. a) Crossover, two parents producing two children. b) Cloning, an individual is passed directly to the next generation. c) Mutation, a single gene (kinetic rate) is changed.

5.4 Case Study: The *E. coli* General Stress Response

In order to examine the effectiveness of our approach, a well documented stochastic model of the *E. coli* σ^{32} stress response pathway was selected. This model has been published in detail previously by Srivastava *et al.* [SPB01] providing the basis for a useful case study. The σ^{32} stress response system of *E. coli* allows the organism to respond to situations that may jeopardise the organism's survival. The response to stress generally involves the coordinated regulation of genes whose products have functions such as protecting essential cellular machinery from damaging environmental factors, facilitating the use of alternate energy sources and inducing the organism to move away from the source of the stress. The coordination of this response is centred around a transcription factor, in this case σ^{32}. The idea is that increased levels of σ^{32} are able to switch on around 30 genes that encode the production of other proteins that alleviate stress, termed σ^{32} induced proteins. Sets of genes that are co-regulated in this fashion are termed *regulons*. Free σ^{32} protein can combine with RNA polymerase (to form $E\sigma^{32}$) to induce the σ^{32} regulon. The level of σ^{32} in the cell is modulated in response to an input to the pathway which senses stressful conditions and induces the production of σ^{32} from its parent gene (*rpoH*). The constant accumulation of σ^{32} is prevented by a protein degradation pathway which is an important regulatory mechanism in this pathway. In *E. coli* this degradation of σ^{32} occurs via a protein produced from the *ftsH* gene. However, in order to be degraded rapidly σ^{32} must be complexed with the protein products of other genes, which are themselves members of the σ^{32} regulon. In this study this complex is refereed to as the J-Comp-σ^{32} complex.

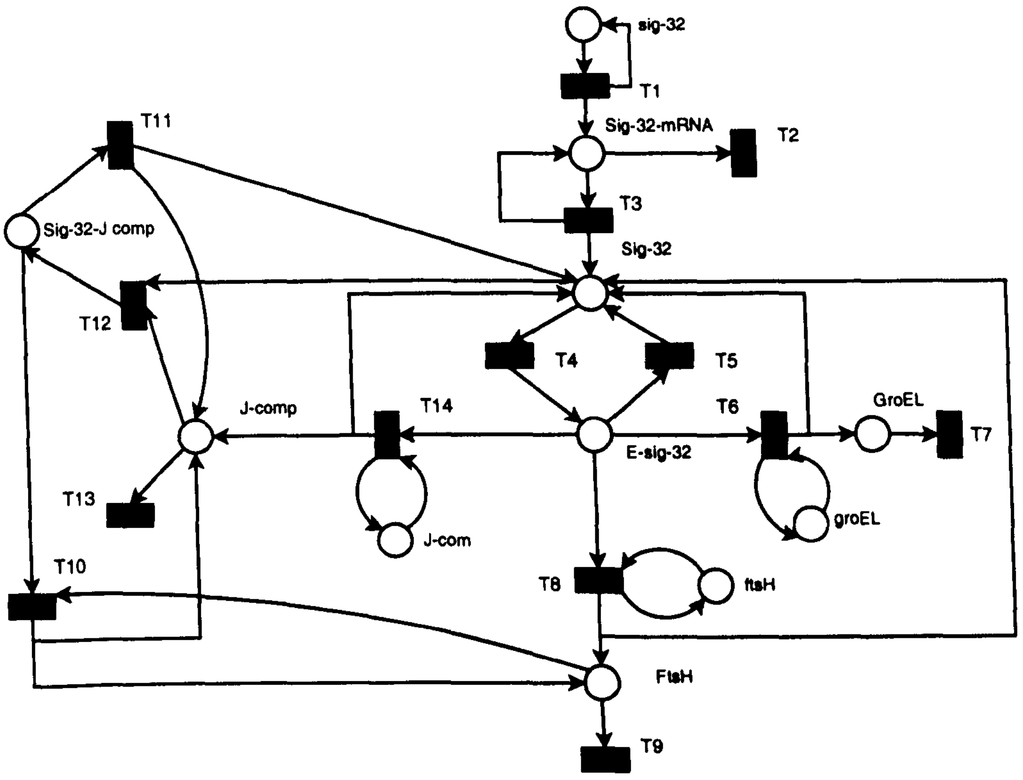


Figure 5.2: The Petri net representation of the σ^{32} stress response pathway of *E. coli*, from Srivastava *et al.* [SPB01]. Initial amounts of molecules were: sig32, groEL, ftsH and jcomp 1; Sig-32 and J-comp 10; Sig-32-mRNA 15; E-Sig-32 25; Sig32-JComp 200; GroEL 270; FtsH 300. The kinetic parameters for the model are listed in [SPB01].

5.4.1 Petri Net Based Modelling of the σ^{32} Stress Response Pathway

The Petri net model of the above regulatory network used to evaluate the genetic algorithms approach is depicted in Figure 5.2, with the corresponding table of transition names in Table 5.1. This is based on the model presented in [SPB01], subject to the alterations described later in this section. Typically when modelling biological systems using SPN's, places represent a particular molecular species, the number of tokens on each place represent the amount of that molecular species present and transitions represent chemical and biological reactions [SKSW04a, SPB01]. The external input to the model is provided by a

Reaction name	Transition #	"gold standard" value
σ^{32} transcription	1	1.4E-5
σ^{32} mRNA decay	2	1.4E-6
σ^{32} translation	3	0.0070
Holoenzyme association	4	0.7
Holoenzyme disassociation	5	0.13
GroEL synthesis	6	0.00472423275181
GroEL degradation	7	1.8E-8
FtsH synthesis	8	0.00366148794999
FtsH degradation	9	7.4E-11
σ^{32} degradation	10	7.4E-11
σ^{32}-J-disassociation	11	4.4E-4
σ^{32}-J-association rate	12	2.7149808609E-4
J-disassociation	13	6.4E-10
J-production	14	0.00366148794999

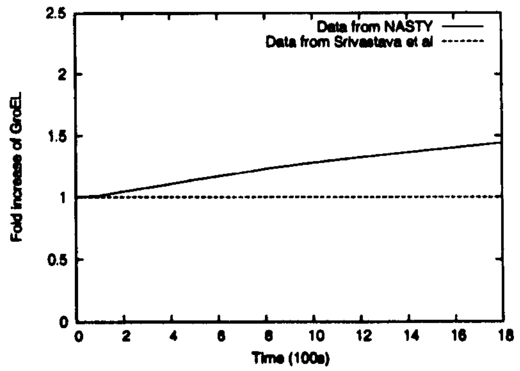
Table 5.1: A listing of the names of the reaction names and the transitions they relate to in Figure 5.2. The "gold-standard" values are the parameters utilised to produce results against which parameter estimation is evaluated.

mechanism that detects stressful conditions, although for clarity, the details of the complex signal transduction systems that act as a sensor mechanism for stress have been abstracted. Essentially, detection of a stressful condition is assumed to alter the rate of the transition $T3$ increasing the production of the σ^{32} protein (labelled Sig-32 in Figure 5.2) through the translation and transcription of the σ^{32} gene (labelled sig-32 in Figure 5.2). The model includes transitions representing the interaction of σ^{32} with RNA polymerase ($T4$ and $T5$), induction of the protein degrading enzyme ftsH ($T8$), and production of the J-complex ($T14$) and association of the J-complex with σ^{32} ($T12$). In our model J-Comp-σ^{32} protein is degraded via transition $T10$. The protein GroEL is a known member of the σ^{32} regulon which is not involved in the direct regulation of σ^{32}. This has been included in the model since the level of GroEL can be used to provide an accurate indication of the induction of the σ^{32} stress response regulon, distinct from the regulation of the regulon.

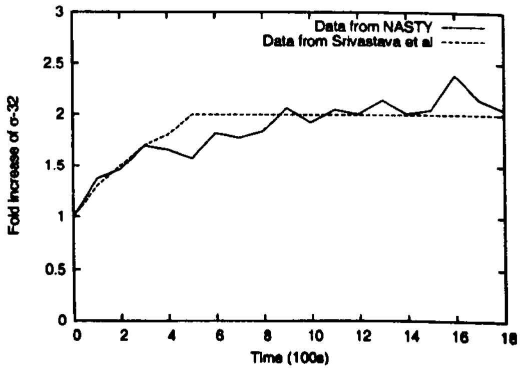
This model of the regulation of the σ^{32} regulon has been employed since it is a realistic model that has been shown to correctly replicate the behaviour of the biological system [SPB01], as determined by laboratory based studies. It is also of a sufficient size and complexity to make a good case study to validate our genetic algorithm. However, in our hands some modifications to the model were required in order to supplement the information given in [SPB01]. The initial concentrations of entities in the model were not explicitly listed, and thus these were estimated from indications given in the paper. In addition, we assume that the DNA and mRNA molecules that encode σ^{32} are outputs of the transcription, $T1$ and translation, $T3$ reactions respectively. With these modifications the behaviour of the model as described by Srivastava [SPB01] could be recreated.

Three particular experiments were selected from those described in [SPB01] to illustrate our approach. These three experiments involved altering the translation rate $T3$. Under no stress $T3 = 0.007$, under anti-sense mediated ethanol stress $T3 = 0.02$ and finally under ethanol stress $T3 = 0.15$. Both σ^{32} and GroEL were monitored under these conditions. The amount of σ^{32} was measured to provide an indication of level of the stress inherent in the pathway. GroEL is a product of the σ^{32} pathway, but is not directly involved in σ^{32} regulation hence the level of GroEL gives an indication of whether the σ^{32} regulon has been induced.

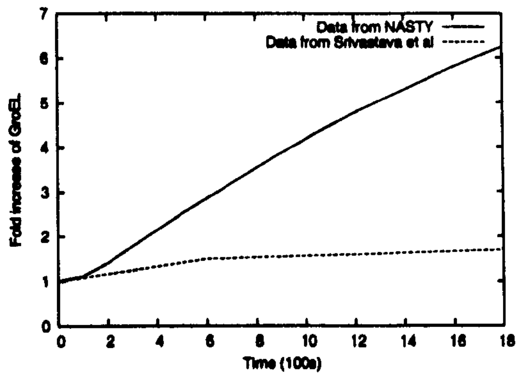
Initially a zero stress situation was simulated in the NASTY tool. The model was simulated 50 times, and the average value was used to compare the fold increase obtained under stress conditions. The results from our simulation under these conditions appear to match well with the results from [SPB01] (data not shown). Next, the model was simulated



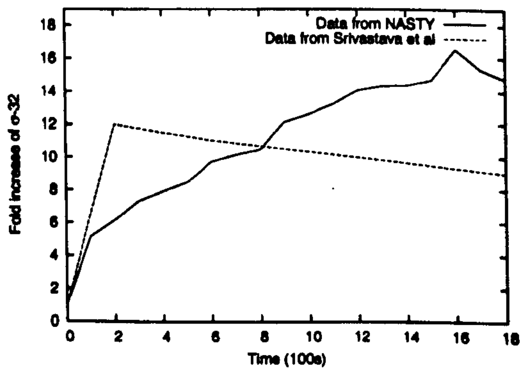
(a) A comparison of GroEL time trajectories under anti-sense mediated stress



(b) A comparison of σ^{32} time trajectories under anti-sense mediated stress



(c) A comparison of GroEL time trajectories under ethanol stress



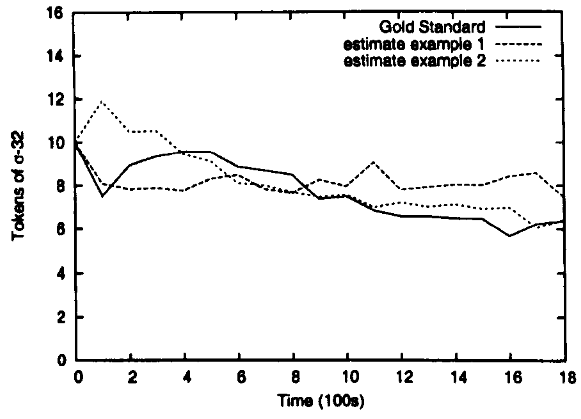
(d) A comparison of σ^{32} time trajectories under ethanol stress

Figure 5.3: The fold increase of token numbers for GroEL and σ^{32} under stress conditions compared to zero stress. Results from NASTY and Srivastava [SPB01]

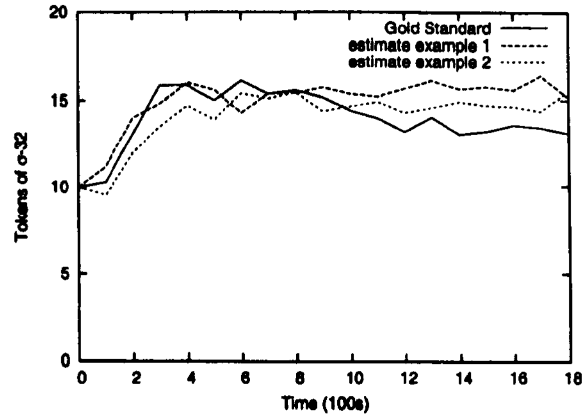
with the translation rate $T3$ adjusted to the levels for anti-sense mediated stress and ethanol stress situations. These results were compared with the results in Srivastava [SPB01], and are shown in Figure 5.3. The results obtained from NASTY for our initially complete stochastic Petri net model are in line with those for the original Srivastava model. Disparities apparent are assumed to be due to the lack of clear initial amounts of molecules detailed in [SPB01]. However, these disparities do not impact on our subsequent studies, since the simulation results obtained from NASTY using our interpretation of the model in [SPB01], are taken forward as the gold standard from which the success of the genetic algorithm is evaluated. The reaction rates representing the gold-standard parameter values are listed in Table 5.1.

5.4.2 Performance With One Time Trajectory

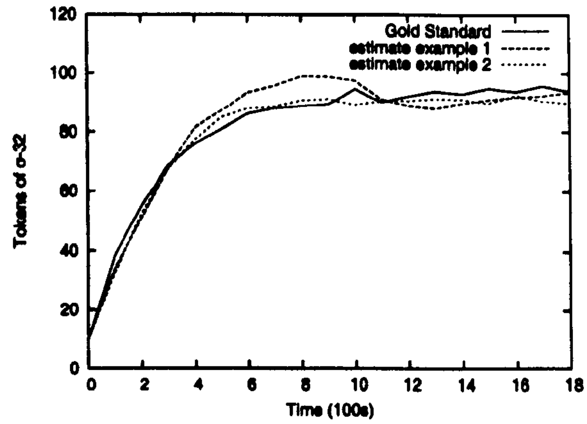
Initially the algorithm was investigated by utilising the input of the time trajectory of one protein, σ^{32}. Only one time trajectory was utilised as typically data on only a handful of proteins is available from laboratory studies. Investigations were carried out to determine the performance of the genetic algorithm under three different stress conditions using the gold standard time trajectories obtained from the NASTY tool. Each run of the algorithm consisted of a population of 2000 individuals over 50 generations of the population. Each individual of each generation was simulated 10 times to obtain an average. This procedure was carried out on a 25-node cluster, with times for each generation taking from approximately 1 to 30 minutes. Initially, experiments were carried out using the time trajectory of a single protein σ^{32}, from the "gold-standard" to evaluate the fitness of solutions in the



(a) σ^{32} under zero stress, p1=0.441, p2=0.001



(b) σ^{32} under anti-sense mediated stress, p1=0.001, p2=0.001



(c) σ^{32} under ethanol stress, p1=0.001, p2=0.001

Figure 5.4: A comparison of selected results obtained from the genetic algorithm, using a single protein's time trajectories (σ^{32}) to evaluate fitness, compared against the gold standard time trajectories (Pearson correlation p values included).

genetic algorithm for each of the three different stress situations. The results for these experiments are shown in Figure 5.4. The obtained parameter values are listed in Table 5.2. When compared to the σ^{32} gold standards, results obtained from the genetic algorithm displayed a highly significant similarity. To determine how well the predicted parameters matched the behaviour of the system more globally, the solutions obtained from the genetic algorithm were compared to the "gold-standard" GroEL time trajectories. These results are shown in Figure 5.5. Results from these comparisons displayed a poor match between the genetic algorithm's solutions and the "gold–standards". This was an interesting and surprising result as the genetic algorithm had found solutions with extremely high fitness, almost maximal, for σ^{32}, whilst being extremely inaccurate in predicting the behaviour of GroEL. The parameters values obtained from the parameter estimation bore very little resemblance to the gold standard parameters. Both these observations indicated that the fitness function may need refinement.

5.4.3 Performance With Two Time Trajectories

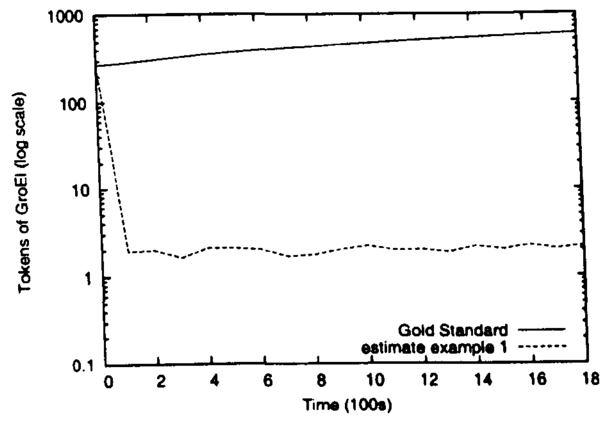
In an attempt to resolve the described situation above, the genetic algorithm was programmed to utilise the "gold-standard" time trajectories for both σ^{32} and GroEL. The gold standard parameter results for the three stress conditions were again used to evaluate the performance of the cost function for the genetic algorithm. The cost function was calculated as in the previous experiment except that scores between 0-1 were calculated for each protein's time trajectory, then added to give a score between 0-2. This experiment was designed to see if the genetic algorithm could be improved by evaluation against the

	Singular		
Transition #	No Stress	Anti-sense stress	Full stress
1	0.02816442236234795	0.6624223051243818	0.6567203171385764
2	0.9892807234664744	0.79024504142124	0.9873173179265117
3	0.1516401901192853	0.1288056454866423	0.41224234571029283
4	0.1911945627920859	0.0486352293781453	0.006257288087781698
5	0.09802299129054048	0.4145813268467678	0.28063691291137915
6	0.7347793164006664	0.6106724058738197	0.43633186663800094
7	0.43971020864520927	0.8241361178127407	0.43411367629326003
8	0.2677126808580925	0.2401881344060749	0.7717283510745174
9	0.31316273018555285	0.7107783097321267	0.9838060981759529
10	0.0021503824373634	0.6510113649057571	0.3237128316843618
11	0.851299854069143	0.5139932604354988	0.8160934955680405
12	0.9416287663822186	0.1148711835334861	0.14277818446264234
13	0.8187942395493835	0.738709527632975	0.8768193119448565
14	0.39345450313963803	0.5593675038370377	0.6147938419035137

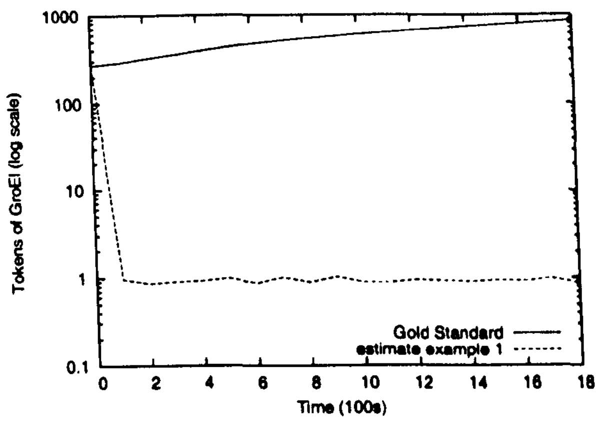
Table 5.2: The parameter values obtained by the genetic algorithm when evaluated against the σ^{32} gold standard results. The results are listed for all three different stress conditions: no stress; anti sense mediated stress and full stress

	Singular		
Transition #	No Stress	Anti-sense stress	Full stress
1	0.592939107894362	0.14148102364021287	0.7979292752631802
2	0.47035445097328354	0.7471702229414248	7.5151261595452E-5
3	0.9913308115176771	0.8009914514574279	0.5474917029181473
4	0.924517749586191	0.12313519448723453	0.09521805298433772
5	0.4525838205268832	0.2703442518479199	0.39532327845469295
6	0.13267041300240534	0.4527362186430416	0.2591654996521209
7	9.273354747080081E-4	4.0376999284197834E-4	0.9401478783147238
8	0.4167693673094178	0.014119743661677675	0.7968081957347457
9	0.8969293435687336	0.934007431211675	0.014055409452780023
10	0.13992699766334615	0.21944862306924395	0.03287876459960182
11	0.47908113015304343	0.5435117124294576	0.6425939057478041
12	0.031027078558680154	0.3466938208494321	0.7787245777904724
13	0.33487754122334934	0.443519980686628	0.5620999519189925
14	0.3455645162146678	0.6869223882493619	0.3754408563331579

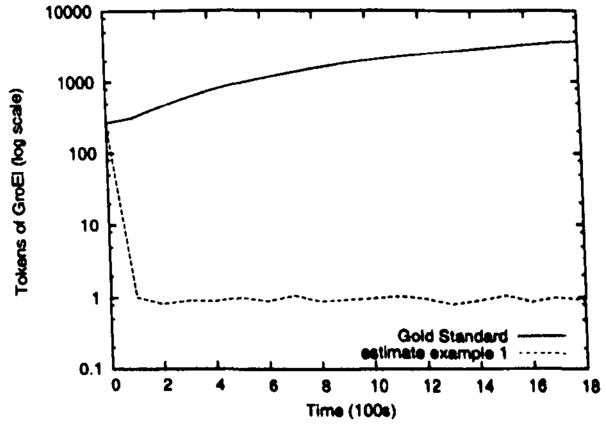
Table 5.3: The parameter values obtained by the genetic algorithm when evaluated against the σ^{32} and GroEL gold standard results. The results are listed for all three different stress conditions: no stress; anti sense mediated stress and full stress



(a) GroEL under no stress, p=0.08

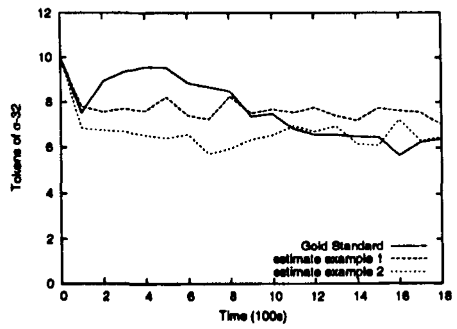


(b) GroEL under anti-sense mediated stress, p=0.959

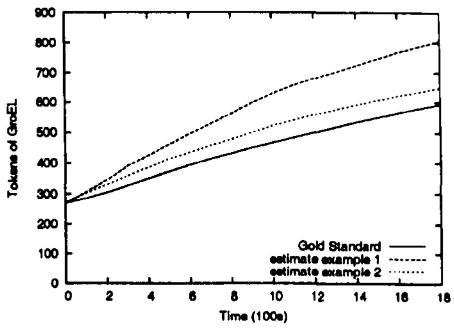


(c) GroEL under ethanol stress, p=0.87

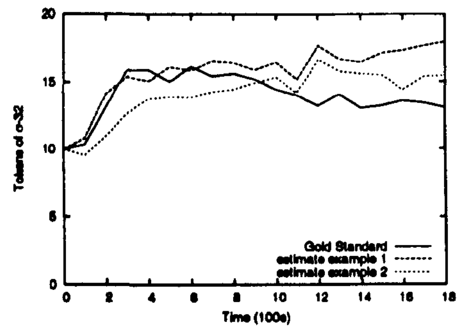
Figure 5.5: A comparison of selected results obtained from the genetic algorithm, using a single protein's time trajectories (σ^{32}) to evaluate fitness, compared against the gold standard time trajectories (Pearson correlation p values included).



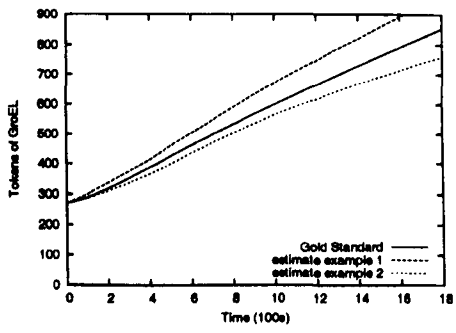
(a) σ^{32} under no stress, p1=0.02, p2=0.27



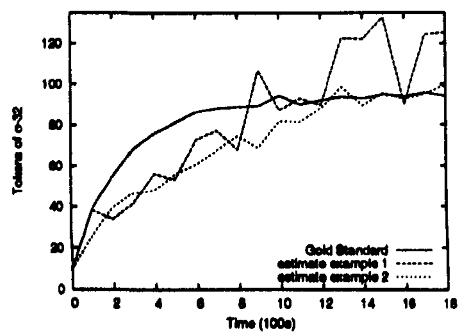
(b) GroEL under no stress, p1=0.001, p2=0.001



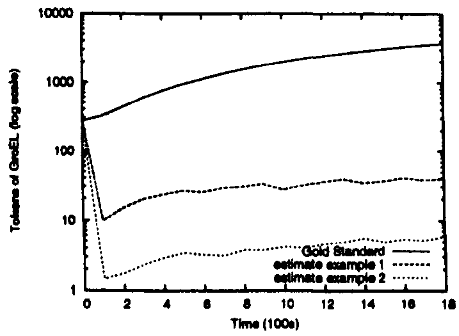
(c) σ^{32} under anti-sense mediated stress, p1=0.001, p2=0.05



(d) GroEL under anti-sense mediated stress, p1=0.001, p2=0.001



(e) σ^{32} under ethanol stress, p1=0.001, p2=0.001



(f) GroEL under ethanol stress, p1=0.193, p2=0.4

Figure 5.6: Solutions obtained from the genetic algorithm using the time trajectories of two proteins (σ^{32} and GroEL) to evaluate fitness, compared with the gold-standards (Pearson correlation p values included for each solution)

time trajectories of two proteins. While time trajectories for every species in the model are highly unlikely to be available from the lab, detailed information for two or three proteins can be more reasonably expected. The estimated parameters obtained by the genetic algorithm were again used to simulate the model, with the results shown in Figure 5.6. The full listing of results can be seen in Table 5.3. The closeness of fit between time trajectories from the estimated individuals with those of the "gold-standard" for σ^{32} were very high, both qualitatively and quantitatively, in all three stress situations. The same was true for the GroEL results for both no stress and anti-sense mediated stress. The exception to these extremely positive results was for GroEL under ethanol stress. Under these conditions the genetic algorithm estimated a suitable solution with regard to σ^{32} in terms of both quantitative and qualitative behaviour. However the solution with regard to GroEL matched qualitatively but was not quantitatively accurate. This indicates that the algorithm is still unable to completely determine the necessary parameters. Techniques for reducing the search space, in order to improve the performance of the genetic algorithm, are required. Again it is noted that the parameters obtained from the genetic algorithm bore little resemblance to the gold standard parameters. The implications of this are discussed in Section 5.6.

5.5 Sensitivity Analysis

In the previous section an attempt to estimate kinetic parameters of a model of the *E. coli* general stress response provided some promising initial results. While the algorithm was successful in most cases, it failed to estimate parameters for a model that reproduced the behaviour of GroEL under ethanol stress, as described by the gold standard. This failure may be attributed, in part, to the large search space resulting from our worst case scenario approach which assumes no existing knowledge about parameters. Also it was assumed that all parameters affected the system equally. Intuitively this is most likely not the case. Thus it appears that further information on parameters, such as parameter bounds, is required to improve the parameter estimation process.

Sensitivity analysis is a simple but powerful technique that is able to give insight into a model by establishing the contribution of an individual kinetic parameter to the emergent behaviour of a complex system [GCPD05, CKW03, IBG+04]. Sensitivity analysis is carried out by varying individual parameter values in a model and observing the resulting behaviour. Small variations to a highly sensitive parameter can drastically change a system's performance and conversely, variations in parameters of low sensitivity generally have little affect. In systems biology, sensitivity analysis has found a number of important applications, see for example [FHC+04, ZGSD03, SGD04].

Classically, sensitivity analysis is applied to a complete model, where a full set of kinetic parameters is assumed to be known. However, in practise, especially early in model development, a fully parameterised model may not always be available. In this section we investigate the use of sensitivity analysis for providing information that can narrow the search space for parameter estimation, concentrating heuristic efforts on the more sensitive parameters and/or constraining parameter values. In particular, the ability of sensitivity analysis to fine tune parameter estimation was evaluated with respect to the following two important questions:

- Which parameters are the most appropriate to focus on?
- What is the range over which the parameter is sensitive to change?

The attempt to parameterise the *E. coli* stress response network, described in Section 5.4, can be viewed as a worst case modelling scenario. Typically there may be some knowledge of parameter values from lab based experiments [GP98, SPB01] which, for example, allows a bound to be imposed on some rates, to reduce the algorithm's search space. Having a bound on a parameter will aid an in-depth sensitivity analysis, allowing the search space to be further reduced, by judging the sensitive areas within pre-placed bounds and allowing a probabilistic optimisation algorithm to concentrating on an even smaller range of values. Thus sensitivity analysis, in combination with parameter bounding, can be utilised in order to direct a probabilistic optimisation algorithm, greatly reducing the search space beyond the initial bounds acquired from laboratory or literature based knowledge.

Sensitivity analysis has previously been applied to continuous deterministic models of biochemical networks. For example, it has been applied to the analysis of the control of oscillatory dynamic cellular processes [GCN+02] and also to investigate the YSNFα-Mediated NF-κB signal transduction pathway [CKW03, IBG+04].

The technique can be applied to nonlinear systems, including signal transduction networks, by the introduction of a sensitivity function [IBG+04, GCPD05]. One commonly used sensitivity function has the following general form [IBG+04]:

$$S_P = \frac{\delta M/M}{\delta P/P} \tag{5.2}$$

where P represents the parameter under investigation and M is the overall response of the system [IBG+04]. It describes the change in M due to the incremental change to the parameter P [CKW03, IBG+04]. This approach does not directly relate to discrete stochastic models, since there is no linear equation describing the overall response of the system that is equivalent to M. Hence approximation methods have to be utilised. The extent of the application of sensitivity analysis to stochastic networks has also been diminished by the computational costs involved [GB01]. Some early work relating to general sensitivity analysis of stochastic systems includes [FHC+04, CS81, DR84]. More recently, Gunawan and co-workers [GCPD05] have investigated techniques for the analysis of discrete stochastic systems described by chemical master equations. This work has significantly influenced the approach developed here.

5.5.1 Strategy and Implementation

In order to carry out sensitivity analysis each rate is individually, systematically altered through a range of values.

There are three main factors that must be taken into consideration:

- Initial values for parameters not under investigation;
- Range and interval of variation for the parameter under investigation;
- How to deal with the compute resources required.

This investigation sought to deduce the usefulness of sensitivity analysis in the context of parameter estimation for a partially described model. As a result there are two distinct application of sensitivity analysis carried out here: those where all network parameters are known and those where parameters are know within a certain bound. The idea is to utilise the results from the former as a gauge of the effectiveness and reliability of the latter results.

In order to facilitate the investigation of a partially described network it was decided that bounds should be placed upon the parameters. The bounds placed upon parameter values take random values in the ranges $10^2 * rate$ to $rate/10^2$. This corresponds to having a "reasonable" idea of the rate. Even in the unknown model, it is key that some value be given to each parameter. Not doing so would require a combinatorial exploration of the state space, this would present an enormous computational challenge. In the test case the rates were exactly as they are in the fully described model.

For both fully and partially described models, each rate is investigated in turn. Each rate under investigation is systematically altered through a logarithmic range from 10^{-14} through to 1.0, with the increment being the current state of the rate multiplied by 1.1. This range fits well with the range given in the original model [SPB01]. For each rate in the sensitivity analysis the network was simulated 10 times with the average time courses for GroEL and σ^{32} being reported and evaluated against the gold-standard utilised in the genetic algorithm. This gives a clear metric of the system's behaviour for each parameter iteration. The metric is then directly utilised in the sensitivity function described by Gunawan *et al.* [GCPD05], namely the finite difference function that directly calculates the change in score over a parameter range.

$$\frac{\delta f}{\delta p_i} = \frac{f(x, p_i + \Delta p_i) - f(x, p_i - \Delta p_i)}{2\Delta p_i} \tag{5.3}$$

This approach clearly entails a large computational effort and while NASTY's job scheduler can address this to some extent, this detailed analysis would still have taken too long to perform. An important point to note is that the simulation jobs required here are, in contrast to the parameter estimation jobs used in the genetic algorithm, known well before the simulation takes place. This allows us to utilise the power of Grid computing in the form of condor [TTL05] to schedule the jobs. Condor is a specialised workload management system for compute-intensive jobs, allowing high throughput computing via the utilisation of an organisations existing compute facilities. Condor seeks to maximise the utilisation of free clock cycles of unused machines. Condor allows the submission and management of contained compute jobs. The NASTY simulation engine and appropriate configuration files were packaged and submitted to the Newcastle University condor pool which contains approximately 1600 nodes. As NASTY was implemented in pure Java, it was possible to utilise Linux, Windows and Mac nodes in the pool. Each stress condition was analysed, with the parameter of each of the 14 transitions in the model being systematically altered. The jobs were submitted to condor, with the jobs taking between thirty minutes to three days to complete.

5.5.2 Results

As detailed in Section 5.5.1, two distinct experimental studies were performed:

1) The application of sensitivity analysis to the *E. coli* model where the complete gold standard parameter set was employed (referred to as the complete model).

2) The application of sensitivity analysis to an *E. coli* model, incorporating random

"fuzzy" parameter rates lying between controlled bounds (referred to as the approximate model).

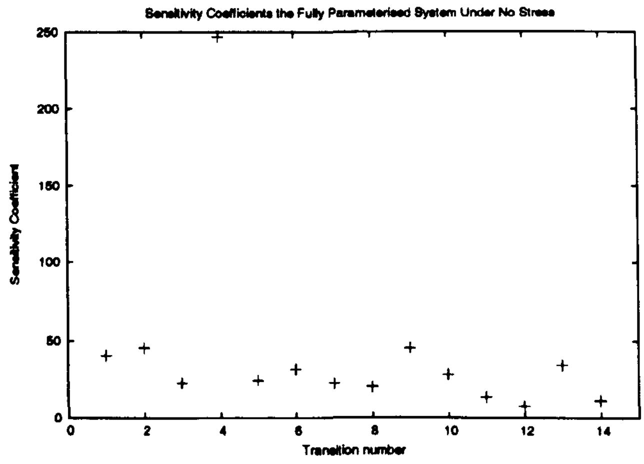
In each study, sensitivity analysis was carried out, systematically varying each rate in turn, and determining the effect on the system by monitoring the production of σ^{32} and GroEL. Each sensitivity experiment was repeated under three stress conditions: no stress; anti-sense mediated stress; and ethanol stress. The aim of this study was to investigate the usefulness of sensitivity analysis in the parameterisation of a model with an imcomplete parameter set, here represented by the approximate model. The study involving the use of a complete model was carried out in order to provide a basis for evaluating the information gained from performing sensitivity analysis on the approximate model.

In this section, a subset of exemplar results are presented and discussed. To aid discussion a table of the reaction names and the transition number they relate to is given in Table 5.1. Again, these transitions all related to the model of the *E. coli* stress response given in Figure 5.2. The full set of results are presented in the appendix.

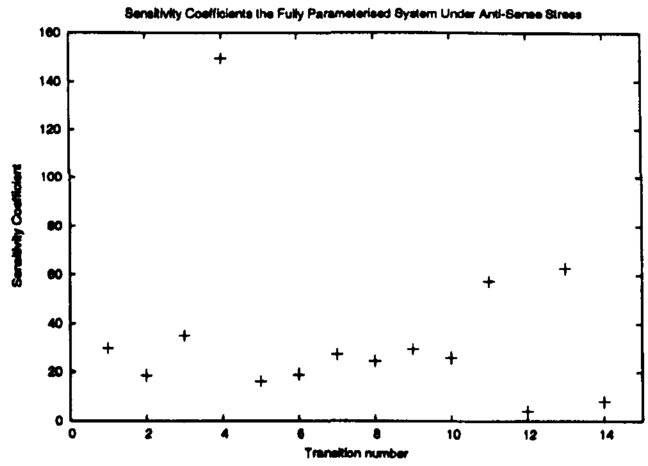
5.5.2.1 Evaluation of the Complete Model

To provide a benchmark as to how sensitivity analysis can given an insight into a model with fuzzy parameters, sensitivity analysis was carried out on the complete model. To obtain a quantifiable value of the sensitivity of the systems dynamics in relation to each parameter in the complete system, a sensitivity coefficient was evaluated for each transition. This coefficient was calculated using the methods from Gunawan *el al* [GCPD05], and described previously. Sensitivity coefficients for each parameter are presented in Figure 5.7.

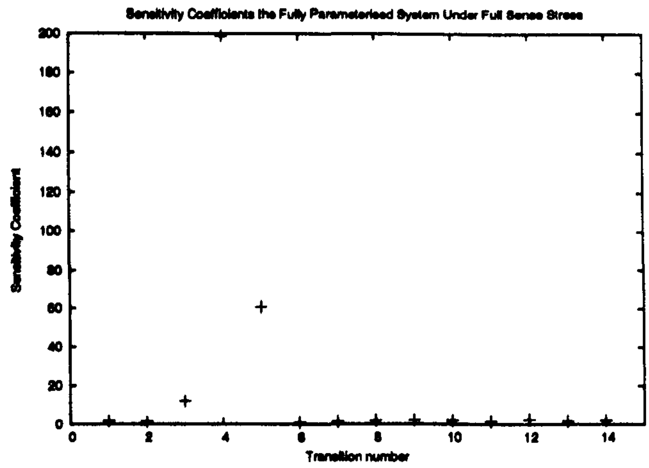
The results in Figure 5.7 demonstrate that the model is highly sensitive to changes in the



(a) Sensitivity coefficient against transition number for conditions of no stress



(b) Sensitivity coefficient against transition number for conditions of anti-sense mediated stress



(c) Sensitivity coefficient against transition number for conditions of full stress

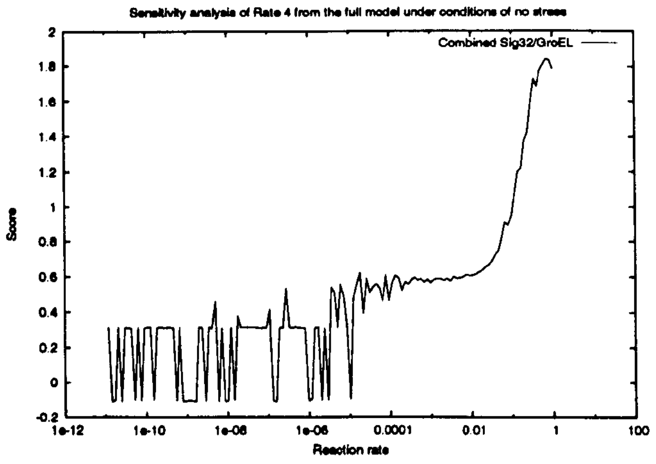
Figure 5.7: The sensitivity coefficients for the complete model. Y axis are not to the same scale

rate of holoenzyme association (transition 4). The effect of varying this rate can be seen in detail by reference to the plots in Figure 5.8 which show the change in the combined score (the cost function utilised in the genetic algorithm) of GroEL and Sig32 (giving a score between 0 and 2) against the rate of translation on a logarithmic scale. Other rate parameters in the model were extremely insensitive when compared to rate parameter 4. Detailed plots for each rate parameter are found in the appendix.

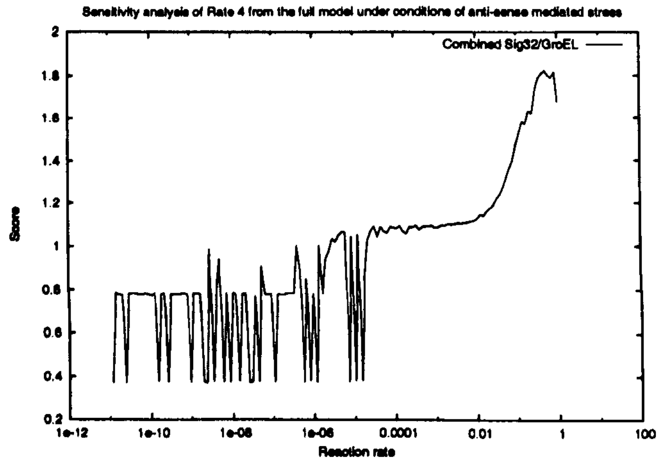
5.5.2.2 Evaluation of the Approximate Model

To determine if sensitivity analysis can aid parameter estimation it is necessary to evaluate the performance with incomplete or "fuzzy" data. Again, to obtain a quantifiable value of the sensitivity of the systems dynamics in relation to each parameter in the approximate system, a sensitivity coefficient was evaluated for each transition using the methods from Gunawan *el al* [GCPD05], and described previously. The results for the approximate model are shown in Figure 5.9. The results presented here varied over a range of values. Some particularly sensitive transitions were observed:

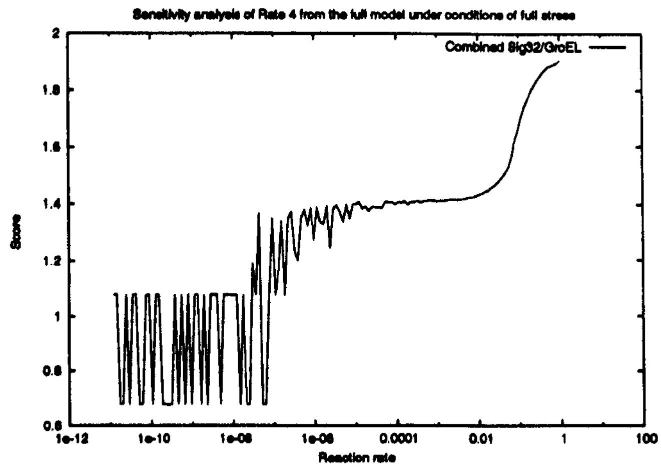
- The high sensitivity of σ^{32}-J-disassociation (transition 11) under anti-sense conditions;
- Very high sensitivity of J-disassociation (transition 13) and high sensitivity of holoenzyme disassociation (transition 5) under full stress conditions;
- Very high sensitivity of σ^{32}-J-disassociation (transition 11) and holoenzyme disassociation (transition 5) in no stress conditions.



(a) Conditions of no stress

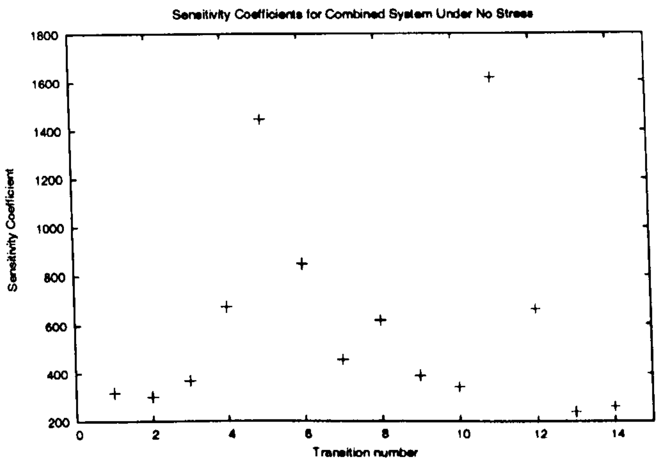


(b) Conditions of anti-sense mediated stress

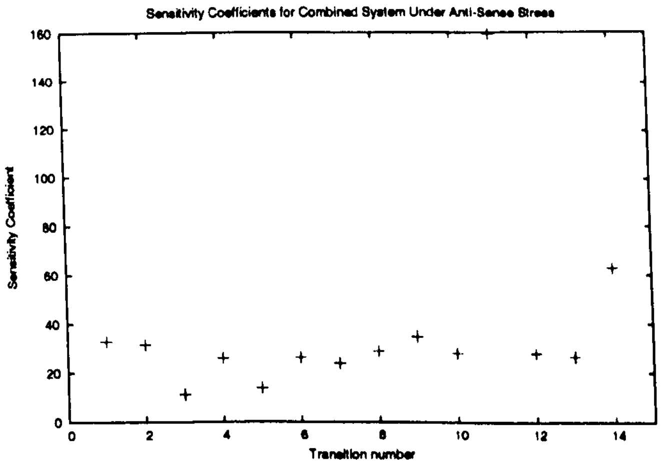


(c) Conditions of full stress

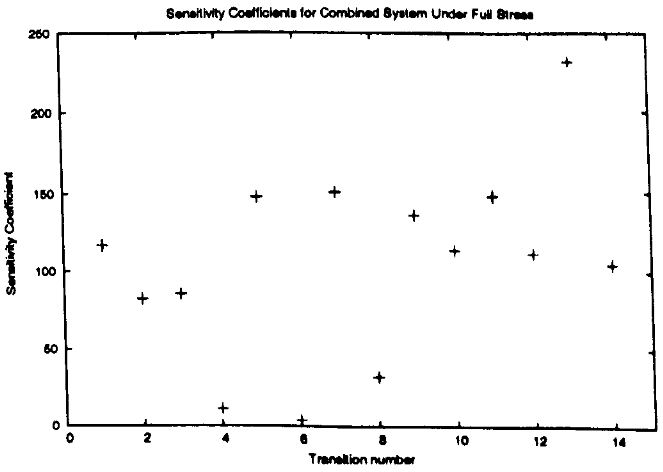
Figure 5.8: Plots of the change in score (the fitness function defined in Section 5.3) for the complete model on a logarithmic scale against reaction rate for both σ^{32} and GroEL for the holoenzyme association rate (parameter 4).



(a) Sensitivity coefficient against transition number for conditions of no stress



(b) Sensitivity coefficient against transition number for conditions of anti-sense mediated stress



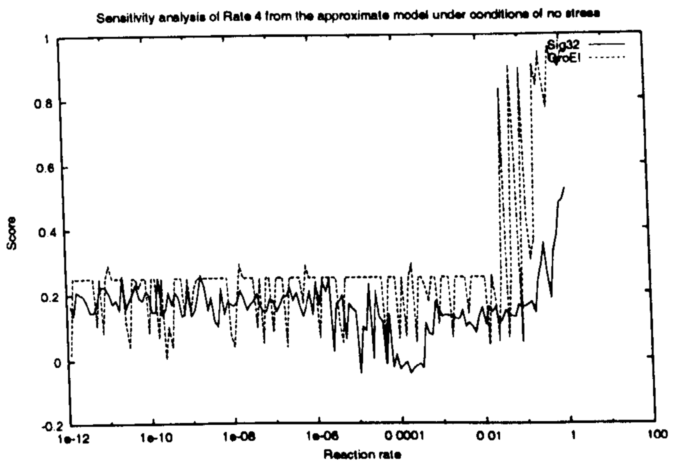
(c) Sensitivity coefficient against transition number for conditions of full stress

Figure 5.9: The sensitivity coefficients for the approximate model. Y axis are not to the same scale.

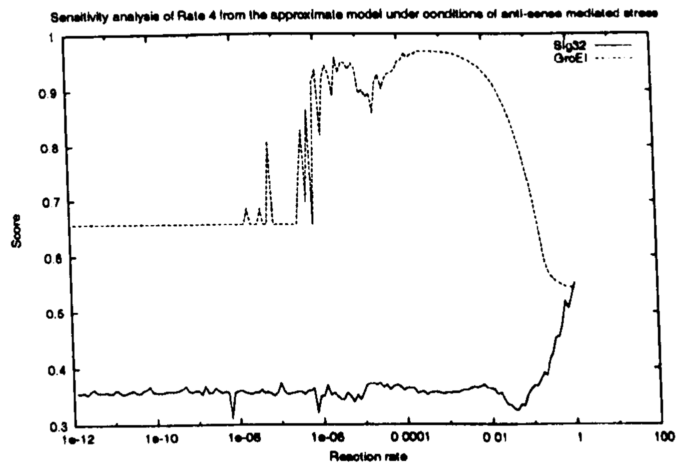
In the complete model it was shown that the system was particularly sensitive to the holoenzyme association rate (reaction 4) and insensitive to other parameters. However, Figure 5.9 suggests that this is not the case for the approximate model.

To investigate this further, we examined the holoenzyme association rate in detail by reference to graphs, presented in Figure 5.10, which show the change in score derived from the fitness function for both σ^{32} and GroEL (calculated in the same manner as in the genetic algorithm presented in Section 5.3) plotted against changes in the σ^{32} holoenzyme association rate (transition 4) on a logarithmic scale. While this parameter appears to be sensitive, it is being masked by other more sensitive parameters. This would make it difficult to correctly direct a parameter estimation algorithm.

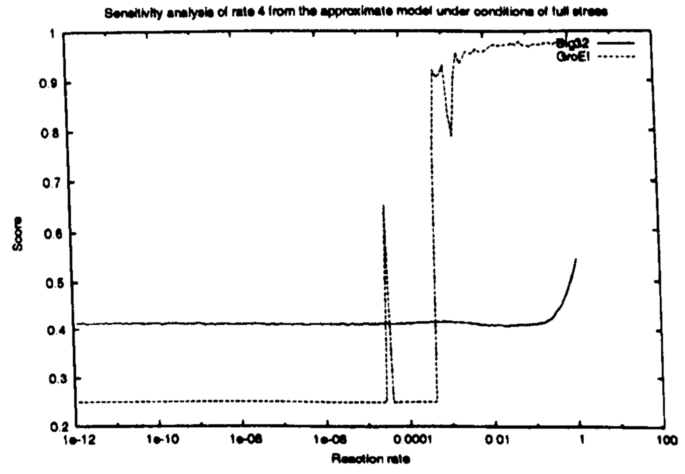
The overall rationale behind this section of work was to investigate the value of sensitivity analysis as a technique for enhancing parameter estimation algorithms, for partially described networks. It is clear from this preliminary study that using sensitivity analysis as a guide for parameter estimation is of limited value since inaccurate rates can lead to very different behaviours for partially parameterised models when compared to the behaviour of the complete system. In this study, the parameters for the rates in the approximate model were randomly assigned within fixed bounds. In future studies it would be interesting to investigate how narrowing these bounds impacts on the usefulness of sensitivity analysis for the purposes of parameter estimation.



(a) Conditions of no stress



(b) Conditions of anti-sense mediated stress



(c) Conditions of full stress

Figure 5.10: Plots of the change in score (the fitness function defined in Section 5.3) for the approximate model on a logarithmic scale against reaction rate for both σ^{32} and GroEL for the holoenzyme association rate.

5.6 Discussion

Stochastic simulations are becoming an increasingly important tool in systems biology. However, the development of stochastic models is hampered by the lack of quantitative data relating to the kinetic parameters which are essential for constructing accurate models [SPB01, GP98]. In this chapter the application of parameter estimation techniques to this problem in the context of stochastic Petri nets has been considered. In particular, a parameter estimation algorithm based on a genetic algorithm has been developed and evaluated by a small case study of stress response system of *E. coli.* The genetic algorithm developed for parameter estimation utilised a simplified view of Darwinian evolution, where a number of solutions are "evolved", based on fitness functions, to a final population, containing individuals with high fitness scores. This procedure relies on the evaluation of a large number of individuals fitness which, in the scope of stochastic simulations, requires the simulation and evaluation of the individual parameter sets in the population.

The NASTY tool presented in Chapter 4 proved to be essential in the implementation of this parameter estimation algorithm. The tool is able to distribute simulation jobs over a cluster of PCs, alleviating the computational cost of simulations [EB01]. The inherent parallelism of the genetic algorithm approach is ideally suited to NASTY's distributed processing, making it more suitable than other more linear heuristics such as simulated annealing [KGjV83]. Previously, the computational cost of this approach has limited its application to parameter estimation [GB01] and thus NASTY was a crucial resource for ensuring the feasibility of the approach. More work is needed to address the problem of the high computational cost associated with stochastic simulations. For example, the use of inexact simulation approaches [Gil01] appear promising ways of reducing simulation times drastically while retaining value for the use in parameter estimation [GW05].

The genetic algorithm based parameter estimation technique was applied to a case study in which the kinetic rates were derived for a stochastic Petri net model of the stress response pathway in the bacterium *E. coli* [SPB01]. The initial results from the case study were promising; using a single protein as the gold standard time trajectory resulted in a parameterised model that correctly replicated the known behaviour of the level of σ^{32}. However, the parameter estimation was less effective for GroEL and results for this protein proved to be inaccurate. To address this problem the genetic algorithm was extended to use the time trajectories for two proteins as the gold-standard for parameterising the network. This appeared to improve the performance of the algorithm allowing the derivation of more accurate rates for the model's behaviour under zero stress, anti-sense mediated stress and ethanol stress. These results raise some important implications for the choice of training data for the parameter estimation algorithm. The parameter values obtained from the genetic algorithm were varied and bore little resemblance to the gold standard data. However the obtained rates were able to reasonably recreate the performance of the gold standard values. This suggests that there are indeed "too many right answers" when considering a model of this type [Kas03]. This has important implication for both model providence and model combination. Combining two smaller biological models into a larger model is unlikely to be successful if "incorrect" parameters are utilised. The models may behave correctly with "incorrect" parameters in isolation, however it will cause problems in combination with other models. To prevent this problem and further aid model parametrisation it is important to know the evidence for parameter values. When creating a biological model it is important that any rates obtained from accurate, intra cellular laboratory experiments, are labelled as such. These parameters should then be held constant through parameter estimation procedures. Holding a small number of parameters is likely to reduce the search space and help direct the genetic algorithm to the "correct" parameters. Hence, the spread of obtained parameters and the inability to fully recreate all behaviours of the system suggested other techniques were required to better understand the system and guide parameter estimation techniques.

While the development of the parameter estimation algorithm was promising, it highlighted the need for complementary approaches to this difficult problem. As a first step, the possibility of using sensitivity analysis [MMB03] to aid in parameter estimation was investigated. The idea was to apply sensitivity analysis to an approximate model and use the insights gained from this to help guide and fine tune the parameter estimation process. This approach again highlighted the inherent computational cost of performing large numbers of stochastic simulations. This was addressed by using a combination of the NASTY simulator and the Condor distributed job scheduling system [TTL05] (allowing access to a resource of close to 2000 processors). This work clearly highlights the significant role that job scheduling systems such as Condor have within stochastic simulation studies.

This application of sensitivity analysis to an approximate model as performed here, did not appear to provide reliable insights which could be used to guide the parameter estimation algorithm. It became clear in our case study that the sensitivity profile of the approximate model based on inaccurate rates contained considerable noise. This was highlighted by the fact the approximate and the complete model displayed very different results. From the complete model it could be seen that the holoenzyme association rate was extremely sensitive, while all other rates were not sensitive. Information like this would greatly help a probabilist parameter estimation algorithm. However this sensitive rate was not found in the analysis of the approximate model. Therefore any indications of how to direct the parameter estimation algorithm would have been spurious. From this preliminary study it appears that care must be taken when using sensitivity analysis to provide information to enhance the parameter estimation process. Sensitivity analysis is however not without merit. Knowing the dynamics of a model may help direct future laboratory experiments by suggesting parameters which are more sensitive to change.

Future work is needed in the area of the application of sensitivity analysis to incomplete models. Our initial study suggests the approach is useful only when dealing with a nearly complete model and that this technique should be applied only in the later stages of the parameter estimation process. However, it would beneficial to establish the degree of precision required in existing rate parameters in order for sensitivity analysis to be of value in the parameter estimation process.

The above research on applying parameter estimation to stochastic Petri net models of biological systems was presented at *Practical Applications of Stochastic Modelling 2005* (PASM'05) and was published in the conferences proceedings [SSW06].

Chapter 6

A Case Study for the Petri Net Framework: The *Bacillus subtilis* Phosphate Stress Response

6.1 Introduction

In previous chapters the development of a number of Petri net [Pet62] tools and techniques has been described, along with their application to the modelling of small, well studied biological systems. These investigations have been designed to provide insight into the applicability of Petri nets to systems biology. Safe Petri net representations of small genetic networks, presented as Boolean networks, have been used to model check reachability properties, thus validating the models (see Section 3.2). P/T net analysis has been applied to metabolic pathways obtained from the KEGG database, demonstrating the utility of Petri nets for validating the basic properties of the pathways through invariant analysis (see Section 3.3). Stochastic Petri nets have been employed to investigate the detailed dynamic behaviour of regulatory systems, with a genetic algorithm employed to automatically parameterise incomplete models.

In this chapter, the scope and limits of a range of these Petri net based techniques is investigated by applying them to a complex, realistic biological modelling case study. The work focuses on developing a model for a novel biological system, namely the phosphate stress response pathway [ASH00] of the soil dwelling bacterium *B.subtilis* [HCW01]. The phosphate stress response pathway in *B. subtilis* is one of a number of regulatory systems that is crucial for the bacterium's survival, as phosphate is the limiting organic compound in the bacterium's natural environment [Hul02]. This chapter investigates the interaction between the different response pathways in the *Pho* regulon, using a previously described study [PH02] as a starting point. In the study, Pragai and co-workers observed hyper-induction of the Phosphate regulon in mutant strains lacking the sigma factor, SigB, and conversely, a hyper-induction of the SigB operon in mutants lacking PhoR. The hypothesis presented in [PH02], is that sigma factor competition is a key factor leading to hyper-induction of the σ^B or Pho regulons in the absence of PhoR or SigB respectively [PH02], a hypothesis which has some support from previous studies [FKN98].

Many computational models of molecular pathways are solely based on developing detailed kinetic models, either stochastic [ARM98], deterministic [IBG+04] or hybrid [MDNM00] in nature. While such studies can provide important insights into the behaviour of the systems under investigation, their construction can often prove problematic due to incomplete knowledge of their structure and the associated kinetic parameters involved (see for example [MS03]). It has been argued that an alternative approach would be to first construct and investigate simpler, qualitative models in order to validate our understanding of the system in question [HK04]. This approach is adopted here. The case study begins by developing a Boolean based safe Petri net model of the Pho stimulon using the methodology presented in Steggles et al. [SBSW05] (see Section 3.2). This Boolean approach is unable to directly model the over expression of a gene (this breaks the Boolean assumption that a gene is simply expressed or not [SBSW05]), so this limitation is overcome by modelling normal expression and over expression of a gene using two different Boolean entities. The resulting model was then analysed utilising the Petri net tool PEP [Gra97], allowing our initial understanding of the regulatory structure and behaviour of the phosphate regulon to be validated before progressing to a more detailed model.

On characterisation of the qualitative model, a preliminary, detailed, stochastic Petri net model of the system was then developed using NASTY (see Section 4.3). The model was constructed using time course data from laboratory studies alongside the insights gained from the Petri net based analysis of the Boolean model. The stochastic model was created in two parts by developing separate models for the Pho regulon and the sigB regulon. The kinetic parameters used in the models were tuned to allow the model to give qualitatively similar results to laboratory data. The resulting model, while clearly not a complete or definitive model, still provides a good basis for further research and also a preliminary insight into the system's behaviour. The preliminary model was investigated using sensitivity analysis methods. This analysis suggests that the binding rates of sigma factors to RNA-polymerase is a key factor in the dynamic behaviour of this model, adding weight to the hypothesis [PH02] that sigma factor competition is a significant factor in the cross talk between these two regulatory networks.

6.2 The Phosphate Regulon and its Interaction with the General Stress Response Regulon.

6.2.1 Phosphate Stress and the Phosphate Stimulon

B. subtilis is subject to a number of stresses in its natural environment, including nutrient deprivation. Phosphate, or more specifically inorganic phosphate (Pi), is a critical limiting nutrient for *B. subtilis* under natural conditions and is probably the limiting nutrient for growth in the soil [Hul02]. The organism has evolved complex regulatory systems for controlling the expression of proteins designed to alleviate Pi stress. The set of genes which have their expression modified due to phosphate stress are referred to as the phosphate stimulon [PAO+04]. The proteins encoded in the phosphate stimulon alleviate stress in a number of ways, such as increasing phosphate uptake and regulating the production and phosphate content of essential cell wall compounds [Hul02].

B. subtilis responds to phosphate stress by controlling the expression of the phosphate stimulon through three pathways, that usually act in combination [PH02]: 1) the general stress response pathway, initiating transcription under the control of the σ^B sigma factor [BVV+97]; 2) the phosphate starvation specific Pho regulon [ASH00, QKH97, PAO+04]; 3) through the induction of PhoP-PhoR/σ^B-independent phosphate starvation genes [PH02].

6.2.2 The Pho Regulon

The genes of the Pho regulon are regulated by the PhoR-PhoP two component signal transduction system [Hul96]. The PhoP response regulator is phosphorylated by its sensor kinase, PhoR. Once phosphorylated, PhoP increases the expression of the PhoPR operon about three fold. Active PhoP also induces or represses other members of the Pho regulon, of which there are currently 31 recognised members [ASH00]. The induction and repression of genes that are controlled by the Pho regulon is controlled by the binding of phosphorylated PhoP to conserved sequences called Pho Box sequences that lie upstream of the coding region. These Pho Box sequences are usually in the form of direct repeats of TOT(A/T/C)ACA with a 3-7bp spacer [ELH99]. Genes in the Pho regulon are generally under the control of the sigma factor SigA, and in a small number of cases YhaX, YhbH, and SigE [PH02].

The Pho regulon is also closely interlinked with a the ResD-ResE signal transduction system, which is required for the Pho regulon to be fully expressed. In a *resD* null mutant the expression of the Pho phenotype is reduced by 80% [Hul02, BLZ$^+$98]. The positive transcriptional regulator AbrB is essential for the remaining 20% of the Pho phenotype, since there is no expression of the Pho phenotype in a *resD/abrB* mutant [Hul02, BLZ$^+$98]. If the phosphate stimulon fails to mitigate the phosphate stress, and the organism is exposed to increasing concentrations of Pi, then the Spo0A response regulator initiates sporulation, terminating the phosphate response by repressing *phoPR* via *resD-resE* and *abrB* [Hul02, BLZ$^+$98]. The Pho regulon and its interactions is depicted in Figure 6.1.

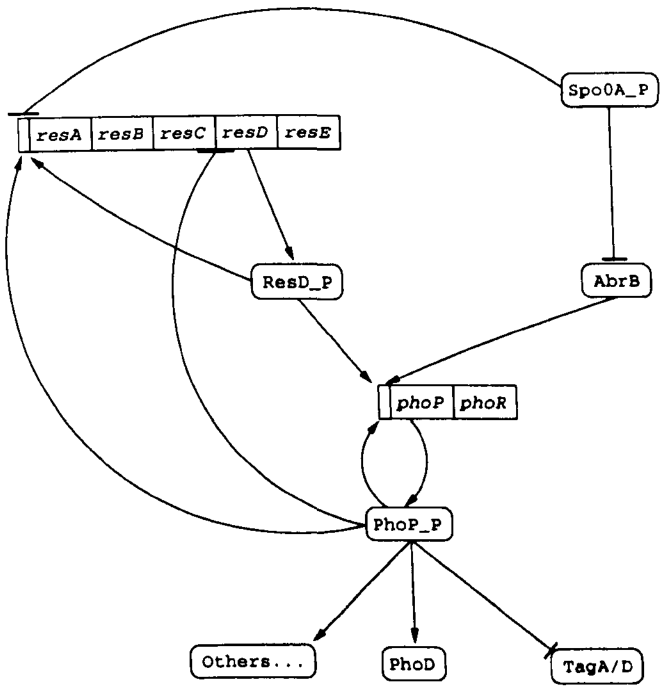


Figure 6.1: A graphical representation of the Pho regulon, taken from [BLZ+98], X_P indicates that a protein X is in a phosphorylated state.

6.2.3 The SigB Regulon

The σ^B general stress response regulon has approximately 200 genes [Hul02]. The general stress response allows *B. subtilis* to deal with both environmental and energy stress. The system is closely linked with the Pho regulon, and both regulons are part of the Pho stimulon [PH02]. Genes encoded by the SigB regulon carry out various functions, including protecting DNA, membranes and proteins [PH02]. The activity of the σ^B regulon is controlled by the anti sigma factor RsbW. When the cell is under stress RsbV is dephosphorylated, resulting in it attacking the Sib-RsbW complex and releasing σ^B.

6.2.4 PhoP-PhoR/σ^B-independent Phosphate Starvation

A small number of genes in the phosphate stimulon are controlled by mechanisms that lie outside of the Pho and σ^B regulons [ASH00]. These do not form a distinct regulatory group and hence were not considered in this study.

6.2.5 Interaction of the σ^B and Pho Regulons

Both the σ^B general stress response regulon and the Pho regulon form part of the Pho stimulon, as interacting subsystems. The interaction of these two regulons is still the subject of much research (see [ANP+04] and [PH02]]). Laboratory studies are have been carried out by studying the phenotypes of mutants in which regulon members and their regulatory components are knocked out singularly, or in combination. For example, a recent investigation aimed at uncovering novel members of the Pho stimulon [PH02], unearthed nine novel members of the Pho stimulon, *yhaX*, hobs, ykoL yttP, ywmG, yheK, ykzA, ysnF and yvgO. Initially the precise regulon controlling the expression of these genes was unknown, but the study went on to decipher this further by the use of knockout experiments, systematically removing σ^B and PhoR by constructing and analysing mutants in which the respective genes had been deleted. The study demonstrated that yhaX, yhbH, ykoL and yttP were under the control of the Pho regulon and that ywmG, yheK, ykzA, ysnF and yvgO were under the control of the general stress regulon [PH02]. This and subsequent, similar studies confirmed that both the σ^B general stress response pathway and the Pho regulon are induced in response to phosphate starvation and that the regulons interact to modulate the level to which each is activated [ASH00, PH02] by showing that the absence of either

σ^B or PhoR leads to increase in expression of the other regulons. These studies therefore provide evidence for a tight coupling of the σ^B general stress response and the Pho regulon [PH02], but currently the exact mechanism for the interaction remains unclear. There is evidence for competition between different sigma factors for the core RNA polymerase enzyme [PH02, FKN98], and data from laboratory studies are consistent with sigma factor competition being responsible for affecting the expression of genes in the Pho regulon. It is envisaged that under conditions where both the Pho and σ^B regulons are operational, there is a competition for the use of a limited pool of the core polymerase. The deletion of one sigma factor by inactivation of its regulon leads to a surplus of the core polymerase and a hyper-induction of the remaining regulon. In support of this hypothesis, Pragai and co-workers [PH02] describe a two-fold enhancement of the transcription of phoPR in a σ^B mutant and an enhancement of the σ^A driven phoPR transcription in σ^H, σ^F, and σ^E mutants [PH02]. Despite these studies, no direct experimental evidence for sigma factor competition has been forthcoming to date, and the possibility of pleitrophic effects caused by inactivation of alternate sigma factors cannot be discounted.

In the following sections, a range of the Petri net based techniques investigated and developed in this project, are applied to modelling and analysing the complex interactions of the σ^B general stress response regulon and the Pho regulon. In particular, the evidence for sigma factor competition hypothesised in the literature [PH02, FKN98] is investigated, and the interaction between these two important pathways is characterised. This case study is a challenging undertaking and is intended to give insight into the scope and limits of the techniques so far considered.

6.3 Boolean Based Petri Net Analysis of the Pho Stimulon

In this section a Boolean based Petri net model of the phosphate stress response pathway for *B. subtilis* [PH02] is developed incrementally, utilising the methodology presented in Steggles et al. [SBSW05] (see Section 3.2). The aim is to construct and validate separately intermediate models for the σ^B and Pho regulons. These models are then integrated by composing them into a third model which captures their key interactions. In this way, a complex Boolean based Petri net model of the Pho stimulon can hopefully be correctly constructed. The resulting Petri net model is then analysed using model checking techniques [Kho03] allowing the complex interactions between the σ^B and Pho regulons to be investigated.

6.3.1 Modelling the σ^B Regulon

One of the mechanisms by which *B. subtilis* can respond to phosphate stress is via the energy stress response which forms part of the general stress response pathway [AGK⁺01]. The end product of this energy stress response is σ^B which transcribes a number of genes to mitigate energy stress [HV98].

There are two distinct stimuli which elicit the general stress response in *B. subtilis*, namely environmental or energy stress [PH02]. In the unstressed state, the anti-sigma factor RsbW sequesters σ^B into a transcriptionally inactive complex. During a period of stress either RsbU (for environmental stress) or RsbP (for energy stress) phosphatases are activated. Both of these are able to dephosphorylate RsbV-P, allowing the product, RsbV, to attack the RsbW-σ^B complex. This results in an anti-anti-sigma factor RsbV-RsbW which is able to free σ^B to complex with RNA polymerase and initiate transcription at least 130 general stress response genes [DDB+04].

The Boolean techniques described in Section 3.2 are employed here to produce a simple qualitative model which captures the signalling pathway based on σ^B by which environmental and energy stress elicit a response from the bacterium. A Boolean network is depicted in Figure 6.2 which captures the essential, high level information about the general stress response σ^B pathway. The truth tables relating to this model are presented in Section G.1. This Boolean network was assembled by extracting the relevant regulatory entities and their associated Boolean truth tables from the literature [AGK+01, HV98]. The tools developed in [SBSW05] can be applied to this network to derive a safe Petri net model of the network, based on asynchronous network semantics, which is amenable to analysis.

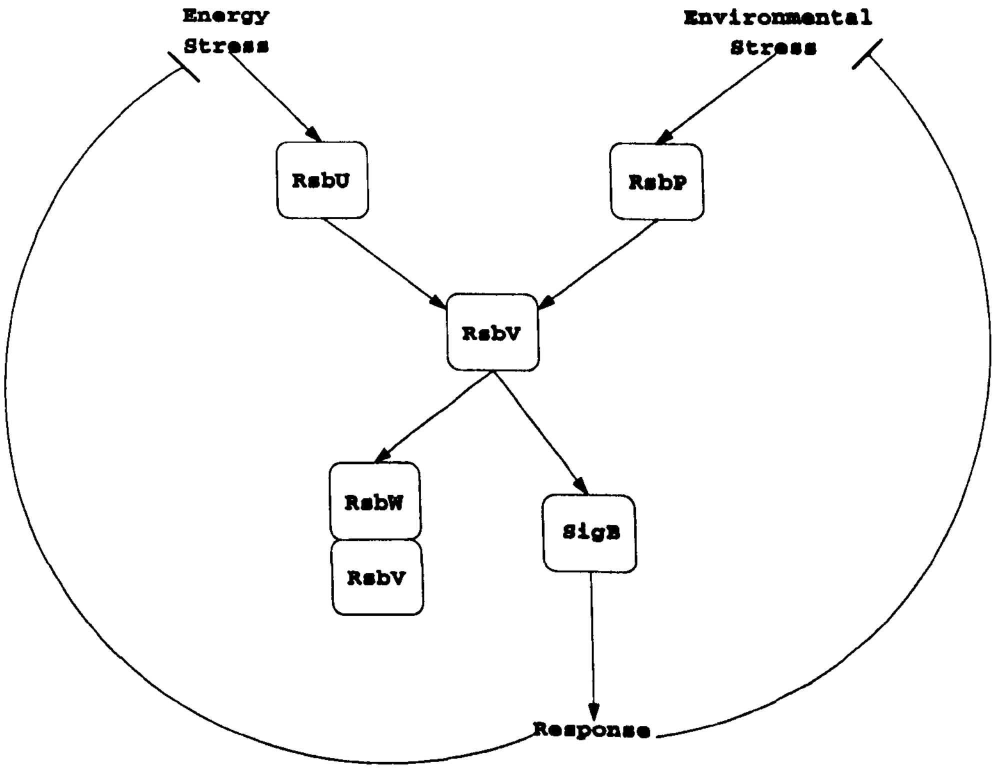


Figure 6.2: A Boolean network representation of the σ^B general stress regulon

Initial Marking		Reachable Marking			
EnvStress	EngStress	Response	EnvStress	EngStress	σ^B
√	√	√	X	X	√
√	X	√	X	X	√
X	√	√	X	X	√
X	X	X	X	X	X

Table 6.1: The reachability results for the Boolean Petri net model of the σ^B general stress response network analysed using PEP.

The model was validated and analysed using the reachability techniques provided by the PEP tool [Gra97]. The results, shown in Table 6.2, validated that the model behaved as expected under environmental and energy stress conditions. In particular, it can be observed that the presence of either stress condition results in the production of RsbV which then attacks the RsbW-σ^B complex. This then releases the Sigma factor σ^B, which, via RNA polymerase, allows the expression of proteins. Once σ^B is released, the induced proteins then act to alleviate the stress condition, producing a stress response. The stress response then acts to reduce the cause of the stress, returning the system to a state of no stress. Conversely, when there is an absence of a phosphate stress condition σ^B is not released, producing no response.

6.3.2 Modelling the Pho Regulon

The next task was to focus on modelling the core of the Pho regulon, depicted in Figure 6.1 (see [BLZ+98]). A Boolean network model was constructed by examining the existing literature [BLZ+98] on the Pho regulon. This indicated that both AbrB and ResD_P enhance the level of PhoP_P, while Spo0A modulates this response by the inhibition of both AbrB and ResD_P [BLZ+98]. This behaviour was modelled by constructing the appropriate truth table definitions for the Boolean network which is presented in Figure 6.3. The truth tables relating to this model are presented in Section G.2. The techniques presented

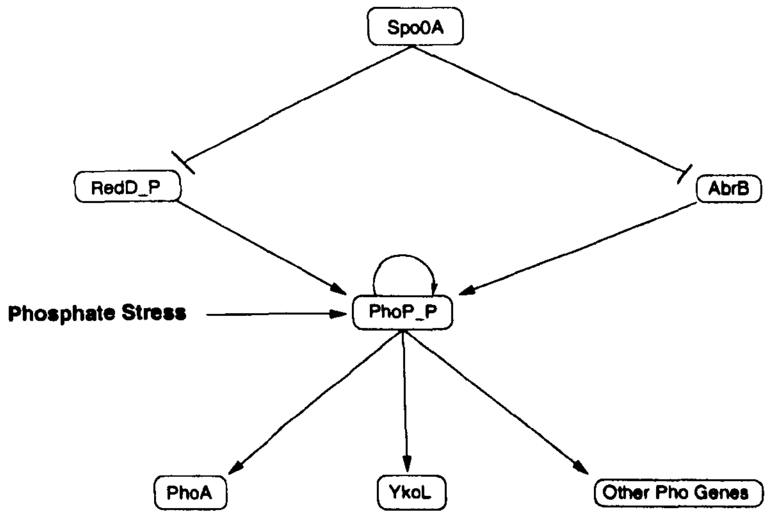


Figure 6.3: A Boolean network representation of the Pho regulon, X_P represents the phosphorylated state of X

in [SBSW05] were then applied again to derive a safe Petri net model of the Pho regulon based on an asynchronous network semantics. This Petri net model was then analysed using the linear programming model checker CLP, developed by V. Khomenko [Kho03]. From the analysis results summarised in Table 6.2, it can be seen that the qualitative Petri net model correctly captures the behaviour of the network and in particular, it confirms that phosphate stress and either of ResD_P or AbrB must be present to activate PhoP_P. Whenever PhoP_P is present the analysis confirmed that ykoL can also be expressed. This results in the production of YkoL and this is taken to be indicative of the activation of the Pho regulon. The model was further analysed to investigate the effect of Spo0A on PhoP_P. In all cases PhoP_P could be inactivated (results not shown), indicating that the model correctly captured the inhibitory effect of Spo0A [BLZ$^+$98].

Initial State				Reachable State	
PhoStress	AbrB	ResD_P	Spo0A	PhoP_P	YkoL
X	X	X	X	X	X
√	X	X	X	X	X
X	√	X	X	X	X
√	√	X	X	√	√
X	X	√	X	X	X
√	X	√	X	√	√
X	√	√	X	X	X
√	√	√	X	√	√
X	X	X	√	X	X
√	X	X	√	X	X
X	√	X	√	X	X
√	√	X	√	√	√
X	X	√	√	X	X
√	X	√	√	√	√
X	√	√	√	X	X
√	√	√	√	√	√

Table 6.2: Model checking results for the Boolean Petri net model of the Pho regulon.

6.3.3 Analysing the Interactions Between the σ^B and Pho Regulons

In the previous sections, intermediate models of the σ^B general stress regulon and the Pho regulon have been constructed and validated. To allow the complex interactions between the σ^B and Pho regulons to be investigated, these two intermediate models need to be correctly combined. To facilitate this a third model was constructed which captured the interactions between these two regulons. This integrated model focused on the behaviour of novel members of the Pho stimulon [PH02], specifically ykoL, a gene under the regulatory control of the Pho regulon and yheK, a gene under the control of the σ^B general stress regulon. Under phosphate stress, these genes are induced normally. However in deletion mutants these genes become hyper induced [PH02]. In particular, in σ^B knockouts, ykoL is hyper-induced while in PhoR knockouts, yheK is hyper-induced. A Boolean model based around the regulatory interactions of these genes in the Pho stimulon is presented in Figure 6.4. The truth tables relating to this model are presented in Section G.3.

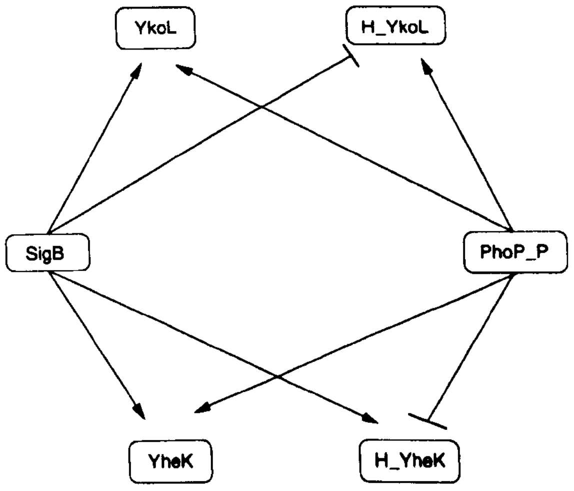


Figure 6.4: A Boolean representation of the conditions resulting in the hyper induction of members of the Pho stimulon. Here H_X represents the hyper-induction of X

Modelling the hyper-induction of a gene requires the use of at least three different states: unexpressed, expressed and hyper induced. This proved to be problematic in a Boolean model since only two different states for each gene are allowed. To address this problem, both Ykol and YheK were given separate Boolean entities to indicate whether they were in hyper induced states. Care was then needed when constructing the model to ensure that biologically invalid states, such as the simultaneous hyper-induction of both YkoL and YkoL, were not reachable. An alternative approach would have been to use a multi-valued network model, an idea which is currently being pursued by a number of groups [SRTC05]. The resulting Boolean model for YkoL and YheK was used to construct a safe Petri net model [SBSW05]. This model was then analysed for coverability properties [Mur89] using the CLP model checking tool [Kho03]. The results are summarised in Figure 6.3. These results validate the behaviour of the safe Petri net model, namely that normal induction of

Current State		Next State			
SigB	PhoP_P	YkoL	H_YkoL	YheK	H_YheK
X	X	X	X	X	X
√	X	X	X	X	√
X	√	X	√	X	X
√	√	√	X	√	X

Table 6.3: Model checking results for the Boolean Petri net model of ykoL and yheK hyper induction

Ykol and YheK is only possible when both the σ^B and Pho regulons are present, and that the genes may be hyper induced when either regulon is removed [PH02].

The above three validated models can now be combined by simply combining their truth table definitions. The truth tables relating to the integrated model are presented in Section G.4. The resulting composite Boolean network model can be processed, using methods described previously [SBSW05] to produce a safe Petri net model of the entire Pho stimulon. This Petri net model was then analysed using the CLP model checking tool [Kho03] and the analytical results are summarised in Table 6.4. It can be seen that under conditions of no stress, no Pho stimulon proteins are induced, as would be expected. When both energy and phosphate stress are present, ykoL and yheK are both induced and hyper-induced. This result is initially counter intuitive. However it must be remembered that the coverability analysis of an asynchronous Petri net model explores all possible reachable states. For example, consider the hyper-induction of ykoL which, according to the literature, should only occur in a σ^B knockout mutant. However, in the Petri net model, many states can be explored before the induction of σ^B, thus resulting in the over expression reported in Table 6.4. Hence a intermediate state can be produced that will be represented in the reachability results.

Current State		Next State					
PhoStress	engStress	YkoL	H_YkoL	YheK	H_YheK	SigB	PhoP_P
X	X	X	X	X	X	X	X
√	X	X	√	X	X	X	√
X	√	X	X	X	√	√	X
√	√	√	√	√	√	√	√

Table 6.4: Reachability results obtained from the complete Boolean model of the Pho stimulon

The hyper induction of ykoL and yheK was further investigated by conducting a range of in silico mutant experiments in which key genes within the Petri net model were knocked out. The idea here was to remove the entity in the truth table for the gene to be knocked out, and to then reconstruct the composite Petri net model, a simple automated task [SBSW05]. The knocked out gene then loses its functional behaviour and simply remains fixed in the state it to which it was initialised. Using this approach, the effect of knocking out σ^B and PhoP_P was investigated. The results of knock-out experiments are presented in Table 6.5 (σ^B knock out) and Table 6.6 (PhoP_P knock out). In the case of the σ^B knockout, the results show that it is impossible to induce the expression of YheK in any form, while YkoL can only be hyper induced. The results for the Pho knockout show it is not possible for YkoL to be induced, while YheK can only be hyper induced.

The behaviour of the model under these knockout experiments replicates high level behaviour seen in laboratory studies [AGK+01, HV98] and thus, further validates the complete qualitative Petri net model constructed in this study. The success of the Boolean Petri net model in correctly replicating the Pho stimulon's behaviour provides an important level of confidence in our understanding of the system. This is extremely important since it provides a solid basis for constructing a detailed stochastic based model of Pho stimulon.

Current State		Next State					
PhoStress	engStress	YkoL	H_YkoL	YheK	H_YheK	SigB	PhoP_P
X	X	X	X	X	X	X	X
√	X	X	√	X	X	X	√
X	√	X	X	X	X	X	X
√	√	X	√	X	X	X	√

Table 6.5: Reachability results obtained from the full Boolean model of the Pho stimulon, with a σ^B knockout

Current State		Next State					
PhoStress	engStress	YkoL	H_YkoL	YheK	H_YheK	SigB	PhoP_P
X	X	X	X	X	X	X	X
√	X	X	X	X	X	X	X
X	√	X	X	X	√	√	X
√	√	X	X	X	√	√	X

Table 6.6: Reachability results obtained from the full Boolean model of the Pho stimulon, with a Pho knockout

6.4 Stochastic Petri Net Analysis

Building on the experience gained from the successful Boolean based modelling study presented in Section 6.3, a quantitative model of the phosphate stimulon was constructed and analysed using stochastic Petri net techniques (see Chapters 4 and 5). The purpose of the model was to investigate the behaviour of the stress response proteins YheK, and YkoL in relation to sigma factor competition for RNA polymerase [PH02]. Such detailed kinetic investigations of biological systems are an important aspect of systems biology, but can be problematic due to the computational time required for stochastic simulations [EB01] and the lack of accurate kinetic parameter data [GP98]. The problem of real time simulation is mitigated here by utilising the NASTY simulator (see Chapter 4). However, the problem of missing kinetic data remains a key problem for this study; despite a number of studies

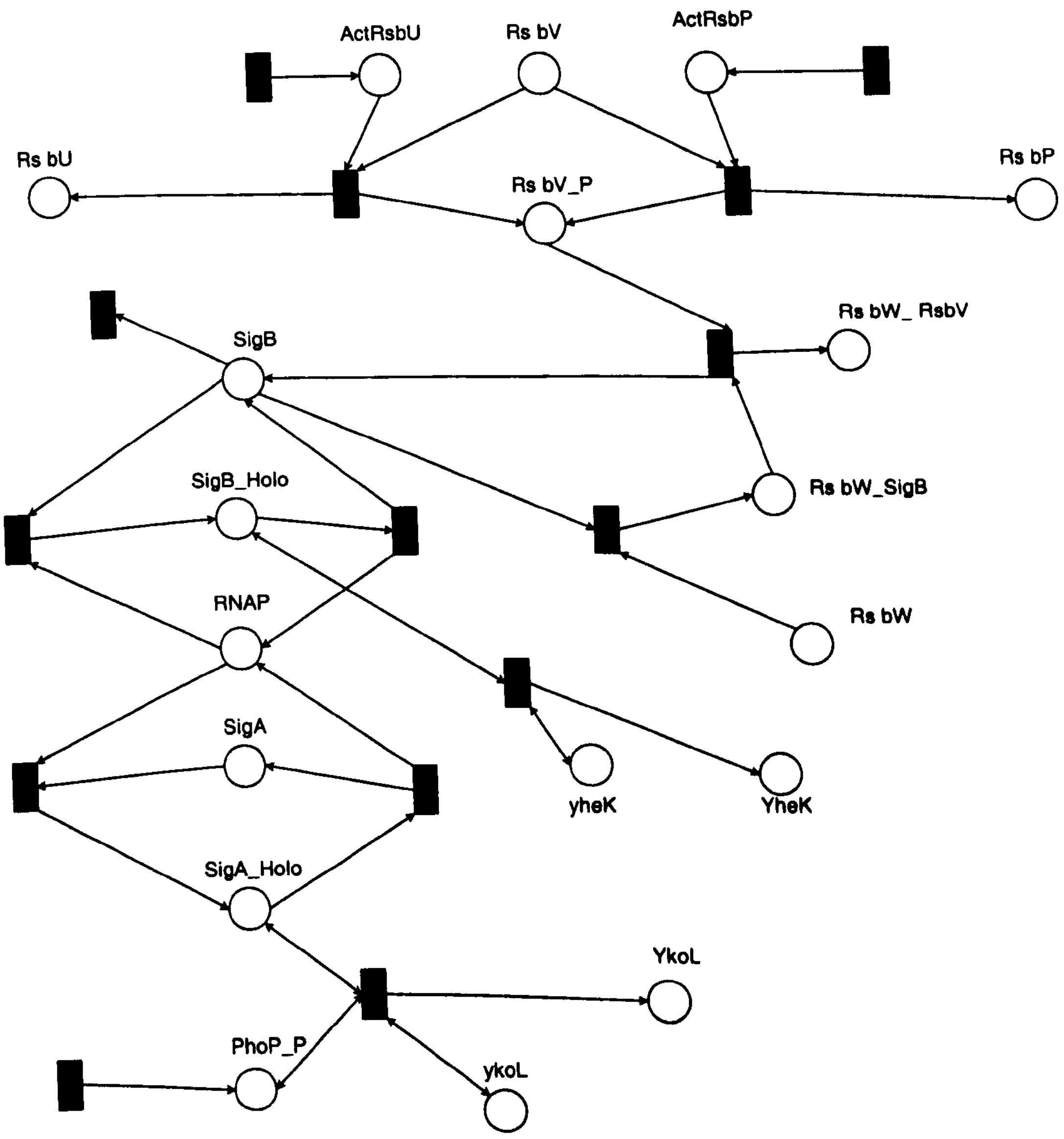


Figure 6.5: A stochastic Petri net representation of the Pho stimulon. Transcription of the σ^B operon is omitted for clarity

that have focused on the molecular binding of molecules in the σ^B general stress response pathway [DLY02], no directly usable rates were available. Thus, this kinetic study of the phosphate stimulon must be considered a preliminary one, in which a high level view of the possible interactions of the system is approximated and studied.

Initially the Pho stimulon's structure was captured by the Petri net model shown in Figure 6.5, which was based on insights gained from the the literature [PH02, DLY02] and the Boolean modelling study undertaken in Section 6.3. The σ^B stress response pathway was modelled in some detail in the Stochastic Petri net model. However, representing the phosphate regulon within the model proved to be problematic due to it being closely interlinked with the underlying cellular machinery via σ^A controlled basal transcription. As a result the phosphate regulon was "black boxed" into a single reaction (see Figure 6.5). The condition of phosphate stress on the system was induced by adding source transitions, PhoP_P and Act_RsbU, to the model.

In an ideal world detailed kinetic rates would be available to assist the construction of an accurate Stochastic Petri net model of the phosphate regulon. Unfortunately, it turns out that in practice there are virtually no kinetic data available to assist the modelling procedure. The aim here therefore is not to create a fully accurate model of the phosphate stimulon, but rather a preliminary model that can be utilised to investigate research hypotheses and guide future experimental studies. The situation is further compounded by a lack of suitable quantitative data with which to train any parameter estimation technique. In particular, the results reported in available laboratory studies are presented in optical density units [PH02], which are extremely difficult to accurately convert to cellular molecular

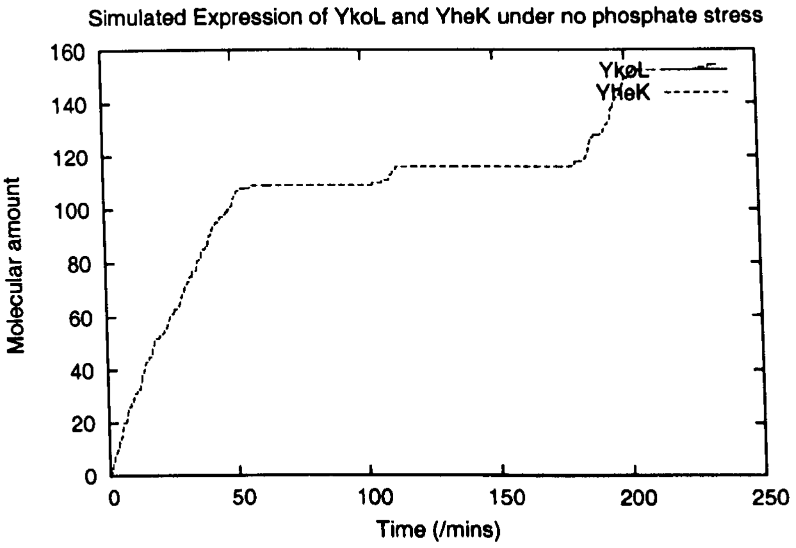


Figure 6.6: The performance of the stochastic model under no stress

amounts. Despite these practical problems it was still possible to manually adjust and manipulate kinetic parameters to allow a useful model to be created which could investigate the effect of sigma factor binding. The approximated kinetic rates appeared to be sufficient for the sensitivity analysis of sigma factor binding, since the model's behaviour qualitatively matched the available laboratory experiments. Simulation results for the resulting stochastic Petri net model under no stress and phosphate stress are presented respectively in Figures 6.6 and 6.7.

These results compared favourably with the results presented in the original *in vivo* study [PH02], and appear to support the approximate model developed here. We next considered analysing the effect of the relative binding rates of RNA polymerase to σ^A and σ^B. The aim of this analysis was to determine if sigma factor competition can account for the hyper induction of that of members of the Pho regulon in a σ^B knockout, and members of the σ^B regulon in a PhoR knockout. To investigate this question the kinetic parameters

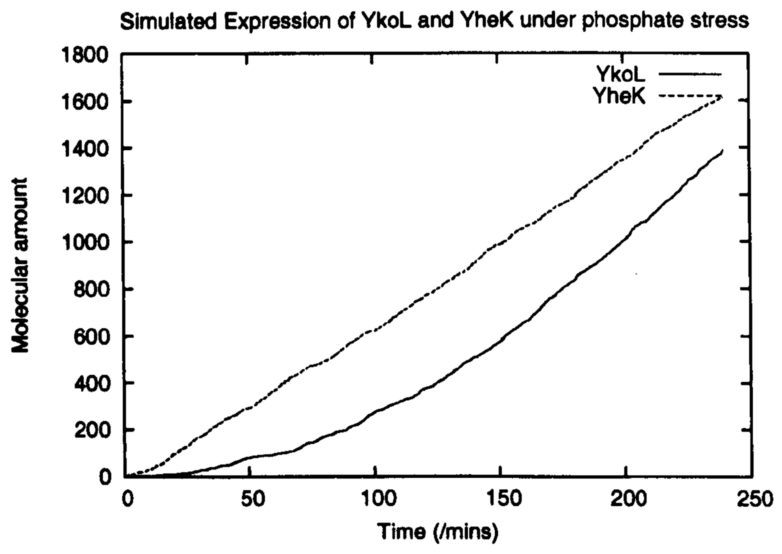


Figure 6.7: The performance of the stochastic model under phosphate stress

Condition	YheK	YkoL
SA*10	0.3	1.3
SA/10	2.3	0.5
SB*10	2.2	0.6
SB/10	0.3	1.3
SA*10SB/10	0.1	1.5
SA/10SB*10	3.7	0.1

Table 6.7: Comparison of the effect of RNA polymerase binding rates on the over expression of members of the Pho and σ^B regulons, relative to the performance of the base model

relating to the binding of RNA polymerase to sigma factors σ^A and σ^B were systematically adjusted by an order of 10. The results of this analysis are presented in Table 6.7 which shows the percentage difference of final protein product for YkoL and YheK for each adjustment of binding kinetics.

The results clearly indicate that the final level of protein product for YkoL and YheK is highly affected by changes to the kinetic binding rates. If RNA polymerase binds more readily with σ^A, then the Pho regulon's expression is greatly increased. If RNA polymerase binds more readily with σ^B then the members of the σ^B regulon are over expressed. These results support the theory of sigma factor competition developed as the result of recent laboratory studies. It has been proposed that strength of the binding of RNA polymerase to Sigma-A, and Sigma-B may depend upon the stress conditions the cell is exposed to [RAK+03]. An iterative set of laboratory and computational studies appears to be required to assist in the confirmation or otherwise of this theory.

6.5 Conclusion

This chapter has sought to investigate the scope and limits of the Petri net techniques presented in this thesis by modelling and analysing a realistic biological example, namely the phosphate stimulon in *B. subtilis* [PH02]. In particular, the study focused on the interactions of the σ^B [HVH05] and Pho regulons [ASH00], with the aim of investigating the experimental hypothesis that sigma factor competition is a key factor leading to hyper-induction of the σ^B or Pho regulons in the absence of PhoR or σ^B respectively [PH02].

To validate an understanding of the system obtained from the literature a Boolean Petri net was constructed using the tools provided in [SBSW05] to model the high level genetic regulatory interactions. This Boolean Petri net model was constructed incrementally by composing three smaller models, which were individually validated using model checking tools [Kho03]. Note that this model construction approach can be seen as providing a useful methodology to aid in the development of complex Boolean Petri net models. Since the initial analysis of the effect of laboratory knockouts had proven promising, the equivalent knockout experiments were carried out in the Boolean Petri net model. This analysis again confirmed that the model's behaviour replicated the high level behaviour presented in the literature, with the hyper-induction of members of the σ^B regulon in the absence of PhoR, and the hyper-induction of members of the Pho regulon in the absence of σ^B.

The construction and validation of the Boolean Petri net model provided an important starting point for the creation of a stochastic Petri net model of the system. Given the lack of data necessary to complete a definitive model, the stochastic Petri net model presented was a preliminary attempt at the kinetic analysis of the Pho stimulon. The model qualitatively matched the behaviour present in the literature well enough to be subject to further investigation. Of particular interest was the effect of sigma factor competition on the expression of members of the Pho stimulon. This was investigated by the sensitivity analysis of the RNA polymerase binding rates. The results obtained showed that these binding rates greatly effected the expression of the genes of the Pho stimulon. If RNA polymerase bound more readily to σ^A, then the Pho regulon was over-expressed; conversely if RNA polymerase bound more readily to σ^B then the σ^B regulon was over-expressed. These results matched well with results reported in laboratory studies, and the sensitivity of binding rates added weight to the hypothesis that sigma factor competition was the root cause of the observed differences in expression.

This study has demonstrated the applicability of Petri net techniques to modelling and analysing a realistic biological study, with Boolean Petri net models validating understanding of the biological system and the stochastic model adding weight to an experimental hypothesis presented in the literature.

Chapter 7

Discussion

Advanced biochemical techniques have allowed studies on a genomic scale, with the ability to track the expression of every gene in a genome over time. The emergence of bioinformatics has allowed scientists to effectively managed this vast data. In combination these techniques have allowed the holistic study of organisms and have lead to development of the field of systems biology. Systems biology studies ideally include an iterative process of lab experiments, data analysis, and modelling of a biological system. A number of supporting technologies must be utilised to analyse data and investigate models of these systems.

In this project Petri nets, a mathematical formalism, have been investigated in the context of computational systems biology. Petri nets have a large body of literature, tools and algorithms and have successfully modelled complex, concurrent networks in the field of computing science [Mur89], manufacturing [ZD90] and hardware design [YK98]. Petri net models can be analysed at a number of different levels depending on the available knowledge of the system to be modelled. If the interactions between genes are known on a simple Boolean interaction basis, then safe Petri nets can be utilised to model the state space of these systems. If the structure of a large biochemical network is known then Petri net structural properties can be investigated, validating the model. Finally if the structure of a system, along with kinetic reaction rates and initial molecular amounts are known then the model can be dynamically simulated, either stochastically or deterministically. This project systematically investigated a number of these techniques in relation to their application to bacterial biochemical and regulatory networks.

7.1 Summary

The volume and variety of data about complex biological systems continues to increase, leading to a new era of data rich biology. In silico modelling is a crucial tool for interpreting these data and generating new experimentally testable hypotheses. Since the pioneering work of Reddy *et al.* [RLM96], Petri nets have been shown to be a potentially important tool for modelling and analysing biological systems. This project has investigated the scope and limits of a range of Petri net techniques applied in this area namely widening the scope of their application by the development of much needed new Petri net technology. The thesis covers a diverse number of sub topics in this area, with particular reference to data interchange standards, qualitative models and stochastic techniques.

One of the advantages of the Petri net framework is the number of tools and techniques available to assist their investigation. Indeed, in the course of the work presented in this thesis, a number of useful existing tools were employed successfully, such as PEP [Gra97] and INA [Sta04]. However, when investigating the application of stochastic Petri nets to studies in systems biology it became clear that there was a deficiency in the available tool support for the specific needs of this research. In particular the lack of a simulator that matched all the needs of the work of this thesis, was apparent. These requirements are summarised below:

- PNML support;
- efficient simulation;
- default use of mass action kinetics;
- computational amenability;
- ability to distribute jobs over a cluster.

To address this shortfall, the NASTY simulator was developed and this proved to be a key tool underpinning the research presented in this thesis. NASTY incorporates an efficient simulation engine based on the amalgamation of the Gibson-Bruck algorithm [GB00] and the underlying structure of a Petri net. NASTY had a job scheduler that allowed computationally intensive simulations to be run over a cluster of machines, mitigating real-time simulation costs. Having full computational access to NASTY allowed the genetic algorithm to be developed, and also allowed the core of NASTY to be utilised to send individual jobs over the condor job management system [TTL05], capturing a vast compute resource. The unique points of the tool mean that it has the potential to make a tangible contribution to similar studies. NASTY will shortly be released as a open source project, will be wrapped as a web service, and will be utilised heavily in the authors long term research goals.

To further the field of systems biology, new and advanced modelling techniques will have to be developed and applied to the investigation of large volumes of data. In order to exploit the vast volume of information, data and models must be available in a computationally amenable form. SBML [HF$^+$03] has arguably become a *de facto* standard for systems biology studies. In order to analyse existing SBML [HF$^+$03] based models with Petri net based techniques which employ the PNML [BCvH$^+$03] standard [BCvH$^+$03], transfer between SBML and PNML is essential. In Chapter 3 conversion between SBML and PNML was discussed and the development of the described mapping and supporting Java tool described. This mapping, and the accompanying Java tool, allows models encoded in SBML to be automatically imported into PNML. This conversion opens a large data resource to Petri net analysis and thus represents an important piece of enabling work. It allowed Petri net techniques to be applied to the analysis of biological networks without the need for the laborious task of creating the models from scratch as Petri nets. To evaluate the mapping two metabolic networks describing glycolysis were validated [TPR$^+$00, HSM$^+$02]. These networks contained information on the topology only and as such were suitable for Petri net based invariant analysis. Models were imported from both the SBML model repository and the KEGG database [KGKN02]. The ability to convert the whole KEGG database to Petri nets was clearly an encouraging development in Petri net modelling.

It is important to have tools and techniques to properly manage the time series simulation data resulting from NASTY. Research into this area failed to identify an existing standard for stochastic time series data. To address this, a time series markup language, TSML, was developed. TSML, a XML based interchange format, was formulated to facilitate efficient storage and analysis of these results. TSML is an efficient, loss-less, data format, that is amenable to computational queries and database storage. As far as the author is aware, TSML is the only results format of its kind currently available. A collaborating project student has developed a support tool for TSML, allowing results to be plotted using the standard UNIX application gnuplot. TSML was utilised alongside NASTY in work described in Chapters 5 and 6, allowing efficient storage and analysis of time series results It is hoped that TSML will grow as a format in its own right, or be taken forward as a sub set of SBML.

The investigation into biological systems via stochastic simulation techniques holds much promise for systems biology studies. Unfortunately this technique suffers from the "perhaps insurmountable" problem of missing kinetic parameters [MS03]. To aid in the construction of stochastic models of biochemical networks, a genetic algorithm for parameter estimation was developed. The genetic algorithm approach presented in Chapter 5 allowed the automatic parameterisation of biochemical networks from very little starting knowledge, namely the model structure, initial conditions and time course data for a small number of genes. The simulation and job distribution powers of NASTY were utilised. NASTY allowed these jobs to be efficiently split across a large number of CPUs, mitigating the real-time cost of this algorithm.

This technique was applied to a previously published model of stress response in *E. coli* [SPB01] and proved successful in estimating suitable parameters. The results of this exercise illustrate the complexity of stochastic networks. Many "right" combinations of parameters [Kas03] were found. A parameter set that was suitable for one particular proteins time trajectory can be completely unsuitable for an others in the same system. This suggests that while heuristic searches are an important tool for stochastic modelling, their results should be treated with care. Parameter estimation techniques should be re-run each time more information arises to ensure the most suitable parameter values are found. This insight will be taken forward in the author's long term research, in which an automated notification service and web-service based workflows will be developed to perform parameter estimation tasks upon the arrival of new, relevant data.

In addition to the genetic algorithm approach, sensitivity analysis was applied to the the *E. coli* stress response network. This work sought to determine how the performance of a model can be apportioned to the dynamics of interacting parameters. Sensitivity analysis was carried out both with the original parameters from an existing model [SPB01] and a random set of "fuzzy" parameters (random values within two orders of magnitude of the original parameters) to determine the effectiveness of sensitivity analysis as a tool to aid parameter value discovery. Each reaction rate was systematically altered and since each alteration requires realisations of the system, the computational costs were high. High throughput computing in the form of condor [TTL05] was utilised, allowing a large volume of compute effort to be carried out over an organisation's unused compute resources. The design of the core of NASTY made this approach possible. To the author's knowledge, this was one of the most extensive sensitivity analysis studies performed on a stochastic system to date. However, further work is needed in order to develop practical parameter estimation techniques for stochastic biological modelling. In particular, the idea of constraining the parameter search space using heuristics based on context specific information seems particularly promising. For example, being able to bound the values considered for a rate parameter by using experimental knowledge could greatly reduce the associated search space. The use of qualitative constraints reflecting the known relationships between genes, such as that the high expression of two particular proteins is mutually exclusive, could also be used to penalise infeasible solutions via the fitness function. A related idea would be to use information about the relative kinetics of a system to fine tune the fitness function. For example, knowing that one reaction occurs much faster than another would again allow infeasible solutions to be removed.

As previously discussed, stochastic simulations of biochemical models provides a promising tool for systems biology. However, there are a number of very real problems associated with this approach. The most pressing is the lack of quantitative data and parameters [MS03]. It is thus also necessary to pursue modelling approaches that do not have such stringent requirements on accuracy of the available data, while still giving insight from the modelling process [KJH05]. In addition, the composition of larger, more complex models from smaller models may prove to be a more effective modelling strategy than constructing them directly. Chapter 3 began by utilising knowledge about some gene regulatory interactions. Literature based knowledge of gene regulation was captured as a series of truth tables. An existing framework allowing a Boolean reduction technique was used [SBSW05] to minimise the truth tables and reduce the state space of the model. The reduced model was then transformed into a safe Petri net via a technique presented by Steggles *et al.* [SBSW05], allowing reachability analysis to be carried out. This Boolean approach was utilised to create a novel model of the well known Lac operon in *E. coli*. Analysis of this model confirmed that its behaviour and construction corresponded with the understanding of the system as described in the literature. The model of the lac operon serves to show how useful models can be created from simply qualitative relationships, demonstrating the validity of the approach for formulating and validating understanding of genetic interactions with a minimal amount of data.

Chapter 6 describes an attempt to consolidate the techniques and developments of this research study by applying them to a novel case study. The system investigated was the phosphate regulatory pathway of *B. subtilis*, which is centred around the Pho regulon. Boolean, simulation, and parameter estimation techniques were applied to the development of a novel in silico model of the Pho regulon in *B. subtilis*. The Boolean based behaviour of the network was investigated first using Petri nets and logic reduction techniques presented in [SBSW05]. A number of small sub-models were constructed and validated, before being combined to form a large Boolean network. Building models via composition of smaller models was shown to be an effective approach, resulting in small, clear test models that can be individually evaluated. However, it was clear that great care must be taken to ensure that nomenclature is standardised to minimise conflicting information. The resulting larger, composite, model was then investigated via reachability analysis [Kho03]. Reachability analysis allows confidence to be gained in the understanding of the system that is captured by the model, by using the Boolean model to validate known behaviours. The Boolean model of the Pho stimulon and the derived analysis, contributed to authors understanding of this system, providing a sound basis for the development of the more complex stochastic model. A number of Boolean based modelling techniques have been described [SH97, AMK00, SBSW05, CRRT04], each applying the techniques to a small case study. The model of the Pho stimulon builds on aspects of these studies, specifically demonstrating the effectiveness of composition of smaller components of the systems and of Petri net based reachability analysis and unfolding.

Once a sound understanding of the Pho regulon had been derived through qualitative modelling, the model was extended and phosphate regulation was further investigated using kinetic modelling. As with many previously described modelling studies there was a problem with the lack of detailed kinetic data. This limited the study to providing a preliminary model of the system. Two sub models of the system were composed, that of the Pho operon and that of the general stress response. These models were then merged to derive a larger model, which was validated and then investigated via the use of brute force sensitivity analysis. This analysis suggested that the balance of expression of different constituents of the Pho stimulon is caused by reactions relating to the competition for RNA polymerase between different sigma factors, supporting a theory put forward by laboratory biologists [PH02]. This case study, while providing insight into the biological system in question, clearly shows the limits and success of modelling at different levels of detail. The Boolean based system produced a reasonably complete, accurate model of the system's behaviour. The stochastic system, however, resulted in a preliminary model that produced results that qualitatively matched experimentally derived results. Sensitivity analysis of this model gave an insight into possible important rates and species. The results matched well with the experimental results described in the literature. Despite the model limitations that can result from a lack of quantitative data for parameter derivation, *in silico* experiments can still provided evidence for behaviour seen in the laboratory.

7.2 Concluding Remarks

The work described in this thesis has demonstrated that the application of Petri nets to biological modelling has the power to form an important part of the emerging field of systems biology. To advance these techniques, fully integrative studies, involving advanced laboratory techniques, statistical interpretation and efficient data management and integration must take place. Indeed this is now occurring, with the creation of a growing number of systems biology centres. It is encouraging to have witnessed the enthusiasm of laboratory biologists, statisticians and theoretical computing scientists about such studies. While advancing the understanding of biological systems, integrated studies clearly benefit the original fields of the constituent experts, both directly (for example, the creation of advanced techniques) and indirectly. For example the experience of deterministic simulation techniques in biologically based studies allowed complex stochastic based voting protocols to become genuinely tractable to analysis [Tho06].

Bibliography

[AGK+01] S. Akbar, T. A. Gaidenko, C. M. Kang, M. O'Reilly, K. M. Devine, and C. W. Price. New family of regulators in the environmental signaling pathway which activates the general stress transcription factor σ^B of *Bacillus subtilis*. *Journal of Bacteriology*, 183(4):1329–1338, 2001.

[AKMM98] T Akutsu, S Kuhara, O Maruyama, and S Miyano. A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions. *Genome Informatics*, 9:151–160, 1998.

[AMK00] T Akutsu, S Miyano, and S Kuhara. Inferring qualitive relations in genetic networks and metabolic pathways. *Bioinformatics*, 16:727–734, 2000.

[ARM98] A. Arkin, J. Ross, and H. H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected *E. coli* cells. *Genetics*, 149:1633–1648, 1998.

[ASH00] H. Antelmann, C. Scharf, and M. Hecker. Phosphate starvation-inducible proteins of *Bacillus subtilis*: Proteomics and transcriptional analysis. *Journal of Bacteriology*, 182(16):4478–4490, 2000.

[BBS05] BBSRC, 15/1/05.

[BCvH$^+$03] J. Billington, S. Christensen, K. van Hee, E. Kindler, O. Kummer, L. Petrucci, R. Post, C. Stehno, and M. Weber. The Petri net markup language: Concepts, technology, and tools. In *Proceedings of the 24th International Conference on Applications and Theory of Petri Nets (ICATPN 2003), Eindhoven, The Netherlands, June 23-27, 2003 — Volume 2679 of Lecture Notes in Computer Science / Wil M. P. van der Aalst and Eike Best (Eds.)*, pages 483–505. Springer-Verlag, June 2003.

[BLZ$^+$98] S. M. Birkey, W. Liu, X. H. Zhang, M. F. Duggan, and F. M. Hulett. Pho signal transduction network reveals direct transcriptional regulation of one two-component system by another two-component regulator: *Bacillus subtilis* PhoP directly regulates production of resd. *Molecular Microbiology*, 30(5):943–953, 1998.

[BMSSV93] R. K. Brayton, P. C. McGeer, J. V. Sanghavi, and A. L. Sangiovanni-Vincentelli. A new exact minimizer for two-level logic synthesis. In Tsutomu Sasao, editor, *Logic Synthesis and Optimization*, pages 1–32. Kluwer, Norwell/MA, USA, 1993.

[BR99] M. Billington, J. Diaz and G. Rozenberg, editors. *Application of Petri Nets to Communication Networks, Advances in Petri Nets*, volume 1605 of *LNCS*. Springer, 1999.

[Bre92] K.J. Breeding. *Digital Design Fundamentals*. Prentice Hall, 1992.

[BVV+97] J. Bernhardt, U. Volker, A. Volker, H. Antelmann, R. Schmid, H. Mach, and M. Hecker. Specific and general stress proteins in *Bacillus subtilis* - a two-dimensional protein electrophoresis study. *Microbiology-Uk*, 143:999–1017, 1997.

[CCD+01] G. Clark, T. Courtney, D. Daly, D. Deavours, S. Derisavi, J. M. Doyle, W. H. Sanders, and Webster P. The mobius modeling tool. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models*, pages 241–250, Aachen, Germany, 2001.

[Cel06] CellML. The CellML webpages at, www.cellml.org, 2006.

[CFKR02] Ming Chen, Andreas Freier, Jacob Köhler, and Alexander Rüegg. The biology Petri net markup language. In *Promise*, pages 150–161, 2002.

[CKW03] S.Y Cho, K.H andShin, W Kolch, and O Wolkenhauer. Experimental design in systems biology based on parameter sensitivity analysis with monte carlo simulation: A case study for the tnfalpha mediated nf-kappab signal transduction pathway. *SIMULATION*, 79:726–739, 2003.

[CRRT04] Claudine Chaouiya, Elisabeth Remy, Paul Ruet, and Denis Thieffry. Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets. In *25th International Conference, ICATPN 2004, Bologna, Italy.*, pages 137–156. Springer-Verlag, September 2004.

[CS81] V. Costanza and J. H. Seinfeld. Stochastic sensitivity analysis in chemical kinetics. *J Chem Phys*, 74:3852–3858, 1981.

[DDB$^+$04] O. Delumeau, S. Dutta, M. Brigulla, G. Kuhnke, S.W. Hardwick, U. Volker, M.D. Yudkin, and R.J. Lewis. Functional and structural characterization of RsbU, a stress signaling protein phosphatase 2C. *J Biol Chem*, 279(39):40927–37, 2004.

[DHP$^+$93] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing*. Chapman and Hall, 1993.

[DK98] J. Desel and E. Kindler. Proving correctness of distributed algorithms using high-level Petri nets - A case study. In *1998 International Conference on Application of Concurrency to System Design, Fukushima, Japan*, pages 177–186. IEEE Computer Society Press, March 1998.

[DLY02] O. Delumeau, R. J. Lewis, and M. D. Yudkin. Protein-protein interactions that regulate the energy stress activation of σ^b in *Bacillus subtilis*. *Journal of Bacteriology*, 184(20):5583–5589, 2002.

[DM01] John Duffy and Paul D. McNelis. Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm. *Journal of Economic Dynamics and Control*, 25(9):1273–1303, 9 2001.

[DMS$^+$05] P.K. Dhar, T.C. Meng, L.Y. Somani, A.K. Sakharkar, A.B.M. Ridwan, S.H.K. Wah, and M. Chitre. Grid cellware: The first grid-enabled tool for modelling and simulating cellular processes. *Bioinformatics*, 21:1284–1287, 2005.

[DR84] D. K. Dacol and H. Rabitz. Sensitivity analysis of stochastic kinetic models. *J Math Phys*, 25:2716–2727, 1984.

[Dre93] T.P. Dreyer. *Modelling With Ordinary Differential Equations*. CRC Press, 1993.

[EB01] D. Endy and R. Brent. Modelling cellular behaviour. *Nature*, 409(6818):391–395, 2001.

[ELH99] S. Eder, W. Liu, and F. M. Hulett. Mutational analysis of the phoD promoter in *Bacillus subtilis*: implications for PhoP binding and promoter activation of Pho regulon promoters. *J Bacteriol*, 181(7):2017–2025, Apr 1999.

[FHC$^+$04] Xiao-Jiang Feng, Sara Hooshangi, David Chen, Genyuan Li, Ron Weiss, and Herschel Rabitz. Optimizing genetic circuits by global sensitivity analysis. *Biophys J*, 87(4):2195–2202, Oct 2004.

[FKN98] A. Farewell, K. Kvint, and T. Nystrom. Negative regulation of RpoS: a case of sigma factor competition. *Molecular Microbiology*, 29:1039–1051, 1998.

[GB00] M.A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry*, 104:1876–1889, 2000.

[GB01] M.A. Gibson and J. Bruck. *Computational Modeling of Genetic and Biochemical Networks*. The MIT Press, 2001.

[GCN$^+$02] Guri Giaever, Angela M Chu, Li Ni, Carla Connelly, Linda Riles, Steeve Véronneau, Sally Dow, Ankuta Lucau-Danila, Keith Anderson, Bruno André, Adam P Arkin, Anna Astromoff, Mohamed El-Bakkoury, Rhonda Bangham, Rocio Benito, Sophie Brachat, Stefano Campanaro, Matt Curtiss, Karen Davis, Adam Deutschbauer, Karl-Dieter Entian, Patrick Flaherty, Francoise Foury, David J Garfinkel, Mark Gerstein, Deanna Gotte, Ulrich Güldener, Johannes H Hegemann, Svenja Hempel, Zelek Herman, Daniel F Jaramillo, Diane E Kelly, Steven L Kelly, Peter Kötter, Darlene LaBonte, David C Lamb, Ning Lan, Hong Liang, Hong Liao, Lucy Liu, Chuanyun Luo, Marc Lussier, Rong Mao, Patrice Menard, Siew Loon Ooi, Jose L Revuelta, Christopher J Roberts, Matthias Rose, Petra Ross-Macdonald, Bart Scherens, Greg Schimmack, Brenda Shafer, Daniel D Shoemaker, Sharon Sookhai-Mahadeo, Reginald K Storms, Jeffrey N Strathern, Giorgio Valle, Marleen Voet, Guido Volckaert, Ching yun Wang, Teresa R Ward, Julie Wilhelmy, Elizabeth A Winzeler, Yonghong Yang, Grace Yen, Elaine Youngman, Kexin Yu, Howard Bussey, Jef D Boeke, Michael Snyder, Peter Philippsen, Ronald W Davis, and Mark Johnston. Functional profiling of the *Saccharomyces cerevisiae* genome. *Nature*, 418(6896):387–391, Jul 2002.

[GCPD05] Rudiyanto Gunawan, Yang Cao, Linda Petzold, and Francis J Doyle. Sensitivity analysis of discrete stochastic systems. *Biophys J*, 88(4):2530–2540, Apr 2005.

[GHS99] I. I. Goryanin, T. Charles Hodgman, and Evgeni Selkov. Mathematical simulation and analysis of cellular metabolism and regulation. *Bioinformatics*, 15(9):749–758, 1999.

[Gib00] M.A. Gibson. *Computational Methods for Stochastic Biological Systems*. PhD thesis, California Inst. Technology, 2000.

[Gil76] D.T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22:403–434, 1976.

[Gil77] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[Gil01] D.T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, 1979.

[GKV01] H. Genrich, R. Kffner, and K. Voss. Executable Petri net models for the analysis of metabolic pathways. *International Journal on Software Tools for Technology Transfer*, 3(4):394–404, 2001.

[GNa06] GNaPN. Gene networks as Petri nets, at http://bioinf.ncl.ac.uk/gnapn, 2006.

[Gol89] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Pub. Co., 1989.

[GP98] P.J.E. Goss and J. Peccoud. Quantitative modelling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National*

Academy of Sciences of the United States of America, 95(12):6750–6755, 1998.

[GPB+04] C. S. Gillespie, C. J. Proctor, D. P. Boys, R. J. andShanley, D. J. Wilkinson, and T. B. L. Kirkwood. A mathematical model of ageing in yeast. *Journal of Theoretical Biology,*, 229(2):189–196, 2004.

[Gra97] Bernd Grahlmann. The pep tool. In Orna Grumberg, editor, *Proceedings of CAV'97 (Computer Aided Verification)*, volume 1254 of *Lecture Notes in Computer Science*, pages 440–443. Springer-Verlag, June 1997.

[GW05] A. Golightly and D.J. Wilkinson. Bayesian inference for stochastic kinetic models using diffusion approximations,. *Biometrics*, 61:781–788, 2005.

[Hac75] M. Hack. *Decidability Questions for Petri Nets.* PhD thesis, Cambridge, Mass.: MIT, Dept. Electrical Engineering, 1975.

[HCW01] C. R. Harwood, S. G. Crawshaw, and A. Wipat. From genome to function: systematic analysis, of the soil bacterium *Bacillus subtilis*. *Comparative and Functional Genomics*, 2(1):22–24, 2001.

[Heb70] P. G. Hebalkar. *Deadlock-Free Sharing of Resources in Asynchronous Systems.* PhD thesis, Cambridge, Mass.: MIT, 1970.

[HF+03] M. Hucka, A. Finney, et al. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

[HK04] Monika Heiner and Ina Koch. Petri net based model validation in systems biology. In *ICATPN*, pages 216–237, 2004.

[HKV01] M. Heiner, I. Koch, and K. Voss. Analysis and simulation of steady states in metabolic pathways with Petri nets. In K. Jensen, editor, *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (CPN'01)*, pages 15–34. Aarhus University, 2001.

[Hof94] R. Hofestadt. A Petri net application to model metabolic processes. *Systems Analysis Modelling Simulation*, 16:113–122, 1994.

[Hol75] J.H. Holland. *Adaptation in natural and artificial systems*. 1975.

[Hol92] J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, 1992.

[HR04] S. Hardy and P.N. Robillard. Modelling and simulation of molecular biology systems using Petri nets: Modelling goals of various approaches. *Journal of Bioinformatics and Computational Biology*, 2:619–637, 2004.

[HSM+02] M.H.N Hoefnagel, M.J.C. Starrenburg, D.E. Martens, J. Hugenholtz, M. Kleerebezem, I.I. Van Swam, R. Bongers, H.V. Westerhoff, and J.L. Snoep. Metabolic engineering of lactic acid bacteria, the combined approach: Kinetic modelling, metabolic control and experimental analysis. *Microbiology*, 148:1003–1013, 2002.

[HT98] Ralf Hofestädt and S. Thelen. Quantitative modeling of biochemical networks. *In Silico Biology*, 1:6, 1998.

[Hul96] F. M. Hulett. The signal-transduction network for Pho regulation in *Bacillus subtilis*. *Molecular Microbiology*, 19(5):933–939, 1996.

[Hul02] F.M. Hulett. *Bacillus subtils and its Closest Relatives, From Genes to Cells*. ASM Press Washington D.C., 2002.

[HV98] M. Hecker and U. Volker. Non-specific, general and multiple stress resistance of growth- restricted *Bacillus subtilis* cells by the expression of the σ^b regulon. *Molecular Microbiology*, 29(5):1129–1136, 1998.

[HVH05] D. Hopper, U. Volker, and M. Hecker. Comprehensive characterization of the contribution of individual SigB-dependent general stress genes to stress resistance of *Bacillus subtilis*. *Journal of Bacteriology*, 187:2810–2826, 2005.

[IBG+04] A.E.C. Ihekwaba, D.S Broomhead, R.L Grimley, N Benson, and D.B Kell. Sensitivity analysis of parameters controlling oscillatory signalling in the nf-kb pathway: the roles of ikk and ikba. *Systems Biology*, 1(1):93–103, 2004.

[IWK+04] R. Iyengar, W. Wolkenhauer, W. Kolch, K. Cho, and U. Kilngmuller. Opening editorial. *Systems Biology*, 1(1):1, 2004.

[JDO04] JDOM. The JDOM home page, at http://www.jdom.org, 2004.

[JM61] F Jacob and J Monod. Genetic regulatory mechanisms in the synthesis of proteins. *J Mol Biol*, 3:318–356, Jun 1961.

[JMBO01] H. Jeong, S. Mason, A.L. Barabasi, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[JNC] JNCC. Joint nature conservation committee.

[Kas03] J. Kastner. *Modeling a Hox Gene Network, Stochastic Simulation with Experimental Perturbation*. PhD thesis, California Institute of Technology, 2003.

[Kau69] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22(3):437–467, 1969.

[Kel04] D. B. Kell. Metabolomics and systems biology: making sense of the soup. *Current Opinion in Microbiology*, 7:296–307, 2004.

[KGjV83] S. Kirkpatrick, C. D. Gelatt jnr, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[KGKN02] M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG databases at GenomeNet. *Nucleic Acids Research*, 30(1):42–46, 2002.

[Kho03] V. Khomenko. *Model Checking Based on Prefixes of Petri Net Unfoldings*. PhD thesis, School of Computing Science, University of Newcastle upon Tyne, 2003.

[Kit01] H. Kitano. *Fundamentals of Systems Biology*. 2001.

[Kit02] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[KJH05] Ina Koch, Bjørn H. Junker, and Monika Heiner. Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics*, 21(7):1219–1226, 2005.

[KO+97] F. Kunst, N. Ogasawara, et al. The complete genome sequence of the gram-positive bacterium *Bacillus subtilis*. *Nature*, pages 249–256, 1997.

[KW01a] Ekkart Kindler and Michael Weber. A universal module concept for Petri nets. an implementation-oriented approach. Informatik-Berichte 150, Humboldt-Universität zu Berlin, April 2001.

[KW01b] K Kindler and M Weber. A universal module concept for Petri nets. In *Proceedings des 8.Workshops Algorithmen und Werkzeuge fur Petrinetze / Gabriel Juhas und Robert Lorenz (Hrsg.) – Katholische Universitat Eichstatt, 2001*, pages 7–12, 1-2 October 2001.

[KZL00] R. Kuffner, R. Zimmer, and T. Lengauer. Pathway analysis in metabolic databases via differential metabolic display (DMD). *Bioinformatics*, 16(9):825–836, 2000.

[LHN04] C.M. Lloyd, M.D.B Halstead, and P.F Nielsen. Cellml: its future, present and past. *Progress in Biophysics and Molecular Biology*, 85:433–450, 2004.

[Lip76] R. J. Lipton. The reachability problem requires exponential space. 62, New Haven, Connecticut: Yale University, Department of Computer Science, Research, January 1976.

[LO03] Kirsten Lenz and Andreas Oberweis. Inter-organizational business process management with XML nets. *Petri Net Technology for Communication-Based Systems — Volume 2472 of Lecture Notes in Computer Science*, pages 243–263, November 2003.

[LQRM02] Chuang Lin, Yang Qu, Fengyuan Ren, and Dan C. Marinescu. Performance equivalent analysis of workflow systems based on stochastic Petri net models. pages 1–64pp, September 2002. InternalNote: Submitted by: hr.

[MA97] H.H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences of the United States of America*, 94(3):814–819, 1997.

[Mar89] M.A. Marsan. Stochastic Petri nets: An elementary introduction. In G. Rozenberg, editor, *Advances in Petri Nets 1989*, pages 1–29. Springer-Verlag, 1989.

[McL01] B. McLaughlin. *Java and XML*. O'Reilly, 2nd edition, 2001.

[MDNM00] H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. *Pacific Symposium on Biocomputing*, 5:338–349, 2000.

[MFD+03] Hiroshi Matsuno, Sachie Fujita, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Towards biopathway modeling and simulation. In *Proceedings of the 24th International Conference on Applications and Theory of Petri Nets (ICATPN 2003), Eindhoven, The Netherlands, June 23-27, 2003 — Volume*

2679 of Lecture Notes in Computer Science / Wil M. P. van der Aalst and Eike Best (Eds.), pages 3–22. Springer-Verlag, June 2003.

[Mit98] I. Mitrani. *Probablisitc Modelling*. Cambridge University Press, 1998.

[MK98] P. Mendes and D.B. Kell. Non-linear optimization of biochemical pathways: applications to metabolic engineering and parameter estimation. *Bioinformatics*, 14(10):869–883, 1998.

[ML97] W. M. Mounts and M. N. Liebman. Qualitative modeling of normal blood coagulation and its pathological states using stochastic activity networks. *International Journal of Biological Macromolecules*, 20(4):265–281, 1997.

[MM99] K. Murphy and S. Mian. Modelling gene expression data using dynamic bayesian networks, 1999.

[MMB03] C.G. Moles, P. Mendes, and J.R. Banga. Parameter estimation in biochemical pathways: A comparison of global optimization methods. *Genome Research*, 13:2567–2474, 2003.

[MNCV00] M. Ajmone Marsan, F. Neri, C. Scarpati Cioffari, and A. Vasco. GSPN models of bridged LAN configurations. *Journal of Systems Architecture*, 46(2):105–130, 2000.

[Mol82] M.K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9):913–917, 1982.

[MS95] H.H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269(5224):650–656, 1995.

[MS03] H.H. McAdams and L. Shapiro. A bacterial cell-cycle regulatory network operating in time and space. *Science*, 301(5641):1874–1877, 2003.

[Mur89] T. Murata. Petri nets - properties, analysis and applications. *Proceedings of the Ieee*, 77(4):541–580, 1989.

[NDMM04] M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. A versatile petri net based architecture for modeling and simulation of complex biological processes. *Genome Informatics*, 15(1):180–197, 2004.

[OV03] Christos A. Ouzounis and Alfonso Valencia. Early bioinformatics: the birth of a discipline - a personal view. *Bioinformatics*, 19(17):2176–2190, 2003.

[PAO$^+$04] Z Prágai, N E Allenby, N O'Connor, S Dubrac, G Rapoport, T Msadek, and C R Harwood. Transcriptional regulation of the phoPR operon in *Bacillus subtilis*. *J Bacteriol*, 186(4):1182–1190, Feb 2004.

[Pet62] C.A Petri. *Kommunikation mit Automaten*. PhD thesis, Bonn, 1962.

[Pet77] J.L. Peterson. Petri nets. *Computing Surveys*, 9(3):223–252, 1977.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1981.

[PH02] Z. Pragai and C. R. Harwood. Regulatory interactions between the Pho and σ^B-dependent general stress regulons of *Bacillus subtilis*. *Microbiology-Sgm*, 148:1593–1602, 2002.

[PNK04] The Petri Net Kernel. The Petri Net Kernel website, http://www.informatik.hu-berlin.de/top/pnk/, 2004.

[PNW04] Petri Nets World. Petri nets world home page, http://www.daimi.au.dk/ petrinet/, 2004.

[PS04] L. Pachter and B. Sturmfels. Parametric inference for biological sequence analysis. *P.N.A.S*, 101(46):16138–16143, 2004.

[PSB$^+$05] C. J. Proctor, C. Soti, R. J. Boys, C. S. Gillespie, D. P. Shanley, D. J. Wilkinson, and T. B. L. Kirkwood. Modelling the actions of chaperones and their role in ageing. *Mechanisms of Ageing and Development*, 126(1):119–131, 2005.

[PWM03] J.W. Pinney, D.R. Westhead, and G.A. McConkey. Petri net representations in systems biology. *Biochem Soc Trans*, 31(6):1513–5, 2003.

[QKH97] Y Qi, Y Kobayashi, and F.M Hulett. The pst operon of *Bacillus subtilis* has a phosphate-regulated promoter and is involved in phosphate transport but not in regulation of the pho regulon. *Journal of Bacteriology*, 179(8):2534–2539, 1997.

[RAK$^+$03] C Rollenhagen, H Antelmann, J Kirstein, O Delumeau, M Hecker, and M.D Yudkin. Binding of sigma(A) and sigma(B) to core RNA polymerase after environmental stress in *Bacillus subtilis*. *Journal of Bacteriology*, 185(1):35–40, 2003.

[RLM96] V. N. Reddy, M. N. Liebman, and M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Computers in Biology and Medicine*, 26(1):9–24, 1996.

[RML93] Venkatramana N. Reddy, Michael L. Mavrovouniotis, and Michael N. Liebman. Petri net representations in metabolic pathways. In *ISMB*, pages 328–336, 1993.

[Ros05] S.M. Ross. *A First Course in Probability*. Prentice Hall, 2005.

[RSJ00] O. Rana, M. Shields, and A. C. Jones. Analyzing java execution semantics using stochastic Petri nets. pages 129–140, 2000.

[SA00] D.B. Strahl and C.D. Allis. The language of covalent histone modifications. *Nature*, 403:41–45, 2000.

[SAS99] Z. Simeu-Abazi and C. Sassine. Maintenance integration in manufacturing systems by using stochastic Petri nets. *International Journal of Production Research*, 37(17):3927–3940, 1999.

[SBM04] SBML. The SBML homepage, at http://www.sbml.org, 2004.

[SBSW05] L.J. Steggles, R. Banks, O. Shaw, and A. Wipat. Constructing and analysis gene regulatory networks using logic simplification and Petri nets. *University of newcastle Tech report*, 2005.

[SGD04] Jörg Stelling, Ernst Dieter Gilles, and Francis J Doyle. Robustness properties of circadian clock architectures. *Proc Natl Acad Sci U S A*, 101(36):13210–13215, Sep 2004.

[SH97] A Silvescu and V Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 11:1–18, 1997.

[SHSW04] O. J. Shaw, Colin Harwood, L. Jason Steggles, and Anil Wipat. SARGE: a tool for creation of putative genetic networks. *Bioinformatics*, 20(18):3638–3640, 2004.

[SKSW04a] O.J. Shaw, A. Koelmans, L.J. Steggles, and A. Wipat. Applying Petri nets to systems biology using XML technology. *In Proceedings of the Workshop on the Definition, Implementation and Application of a Standard Interchange Format for Petri Nets, Satellite event of the 25th International Conference on Application and Theory of Petri Nets, Bologna, Italy*, pages 11–25, 2004.

[SKSW04b] O.J. Shaw, A. Koelmans, L.J. Steggles, and A. Wipat. Applying Petri nets to systems biology using XML technology. Technical Report CS-TR-827, University of Newcastle UK, 2004.

[SM07] Sun-Microsystems. The java homepage, at http://www.sun.java.com, 2007.

[SPB01] R. Srivastava, M.S. Peterson, and W.E. Bentley. Stochastic kinetic analysis of the *Escherichia coli* stress circuit using sigma(32)-targeted antisense. *Biotechnology and Bioengineering*, 75(1):120–129, 2001.

[SPM$^+$02] S. Schuster, T. Pfeiffer, F. Moldenhauer, I. Koch, and T. Dandekar. Exploring the pathway structure of metabolism: decomposition into subnetworks and application to *Mycoplasma pneumoniae*. *Bioinformatics*, 18(2):351–361, 2002.

[SRTC05] E. Simao, E. Remy, D. Thieffry, and C. Chaouiya. Qualitative modelling of regulated metabolic pathways: Application to the tryptophan biosynthesis in *E. coli*. *Bioinformatics*, 21(suppl2):ii190–ii196, 2005.

[SSDB95] M Schena, D Shalon, R W Davis, and P O Brown. Quantitive monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270:467–470, 1995.

[SSW06] O.J. Shaw, L.J. Steggles, and A. Wipat. Automatic parameterisation of stochastic petri net models of biological networks. *ENTCS*, 13:111–129, 2006.

[Sta04] P. H. Starke. INA Integrated Net Analyzer version 2.2, at http://www.informatik.hu-berlin.de/lehrstuehle/automaten/ina, 2004.

[SYSY02] R. Srivastava, L. You, J. Summers, and J. Yin. Stochastic vs. deterministic modeling of intracellular viral kinetics. *J Theor Biol.*, 218(3):309–321, 2002.

[Tho06] Nigel Thomas. Approximation in non-product form finite capacity queue systems. *Future Gener. Comput. Syst.*, 22(7):820–827, 2006.

[TL02] D. Tsavachidou and M. N. Liebman. Modeling and simulation of pathways in menopause. *Journal of the American Medical Informatics Association*, 9:461–471, 2002.

[TPR$^+$00] B. Teusink, J. Passarge, C. A. Reijenga, E. Esgalhado, C. C. van der Weijden, M. Schepper, M. C. Walsh, B. M. Bakker, K. van Dam, H. V. Westerhoff, and J. L. Snoep. Can yeast glycolysis be understood in terms of in vitro kinetics of the constituent enzymes? Testing biochemistry. *European Journal of Biochemistry*, 267(17):5313–5329, 2000.

[TTL05] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[W3C04] W3C. W3C's math home page, at http://www.w3.org/math, 2004.

[Wil06] D. Wilkinson. *Stochastic Modelling for Systems Biology*. Chapman & Hall, 2006.

[WK02] M. Weber and E. Kindler. The Petri Net Markup Language. In H. Ehrig, W. Reisig, G. Rozenberg, and H. Weber, editors, *Petri Net Technology for Communication Based Systems*, volume to appear. Springer, 2002.

[WPT02] D.W Westhead, J.H Parish, and R.M Twyman. *Bioinformatics*. BIOS Scientific Publishers, 2002.

[YK98] A. V. Yakovlev and A. M. Koelmans. Petri nets and digital hardware design. *Lecture Notes in Computer Science: Lectures on Petri Nets II: Applications*, 1492, 1998.

[Zar88] M. Zargham. Parallel processing neural networks. In *Proceedings of the First Annual Meeting of the International Neural Network Society, 1988, Boston, MA, USA; Neural Networks, Suppl.*, pages 1–558 pp., 1988. NewsletterInfo: 34Published as Proceedings of the First Annual Meeting of the International Neural Network Society, 1988, Boston, MA, USA; Neural Networks, Suppl., volume 1, number 1.

[ZD90] M. Zhou and F. DiCesare. Modeling buffers in automated manufacturing systems using Petri nets. In *Proceedings of Rensselaer's Second International Conference on Computer Integrated Manufacturing, 1990, Troy, NY, USA*, pages 265–272, Los Alamitos, CA, USA, 1990. IEEE Comput. Soc. Press.

[ZGSD03] Daniel E Zak, Gregory E Gonye, James S Schwaber, and Francis J Doyle. Importance of input perturbations and stochastic gene expression in the reverse engineering of genetic regulatory networks: insights from an identifiability analysis of an in silico network. *Genome Res*, 13(11):2396–2405, Nov 2003.

[ZL02] S. Zolfraghari and M. Liang. Comparative study of simulated annealing, genetic algorithms and tabu search for solving comprehensive machinegrouping problems. *International Journal of Production Research*, 40:2141–2158, 2002.

Appendix A

Sensitivity Analysis of the Full *E. coli* model under conditions of full stress

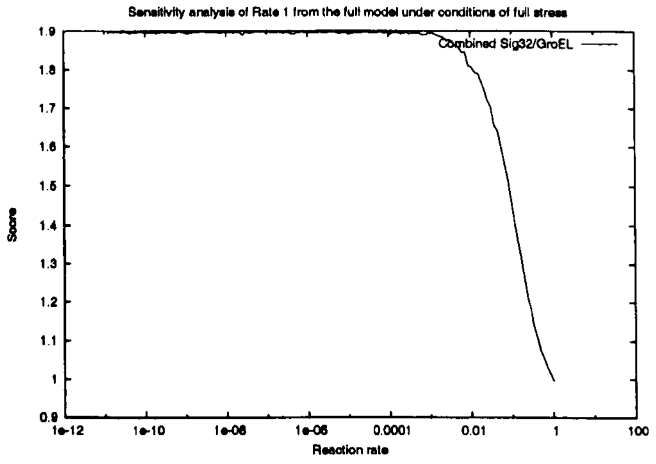


Figure A.1: Sensitivity analysis of Transition 1 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

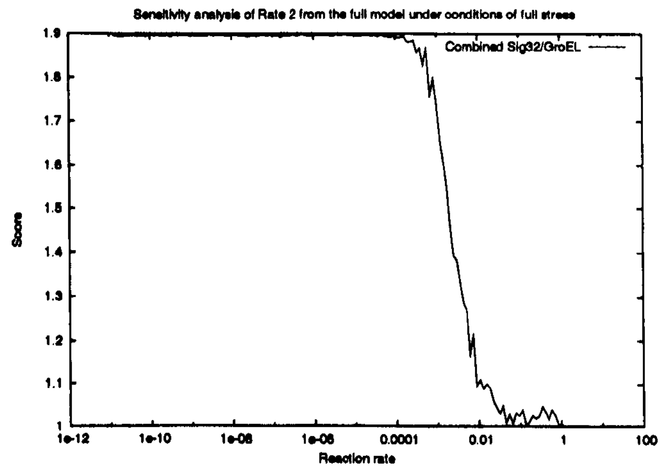


Figure A.2: Sensitivity analysis of Transition 2 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

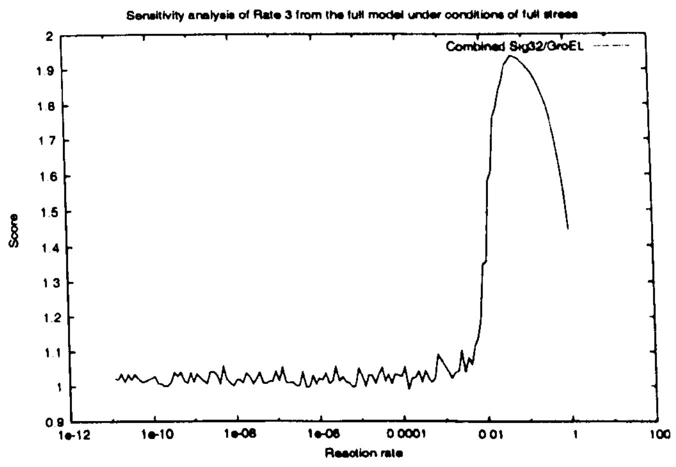


Figure A.3: Sensitivity analysis of Transition 3 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

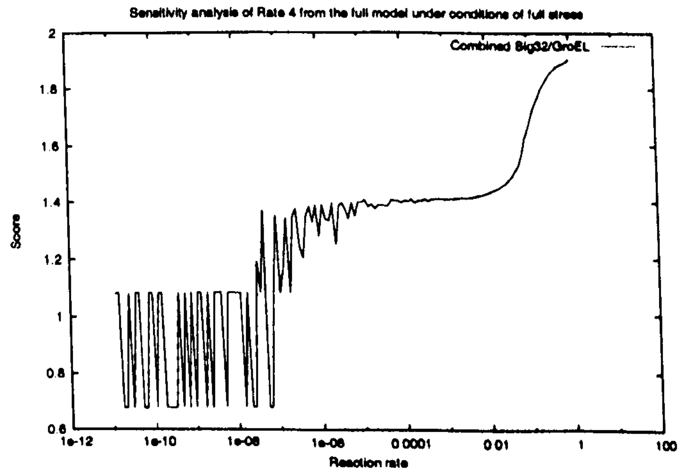


Figure A.4: Sensitivity analysis of Transition 4 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

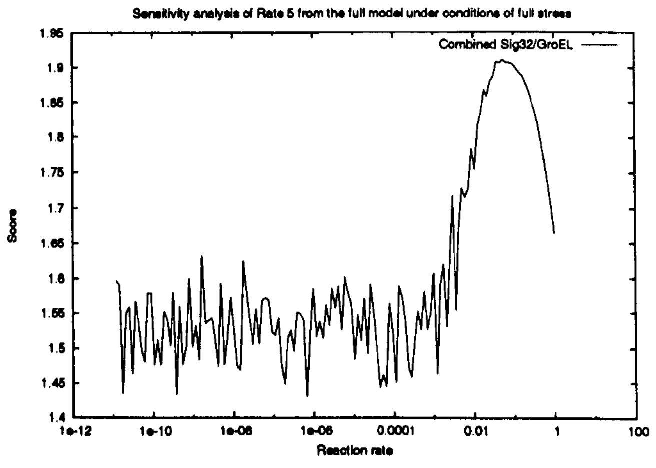


Figure A.5: Sensitivity analysis of Transition 5 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

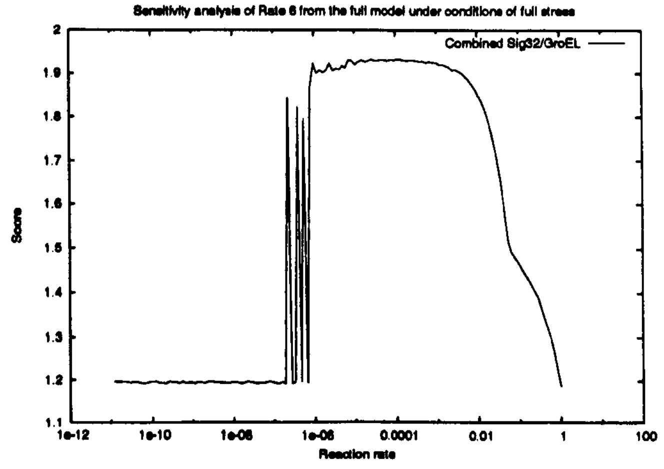


Figure A.6: Sensitivity analysis of Transition 6 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

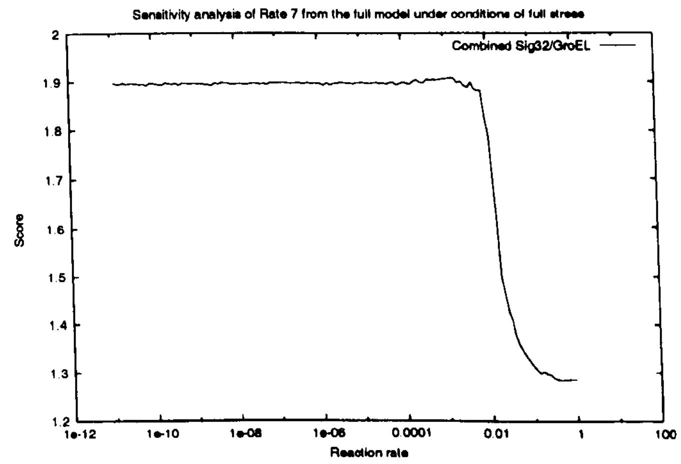


Figure A.7: Sensitivity analysis of Transition 7 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

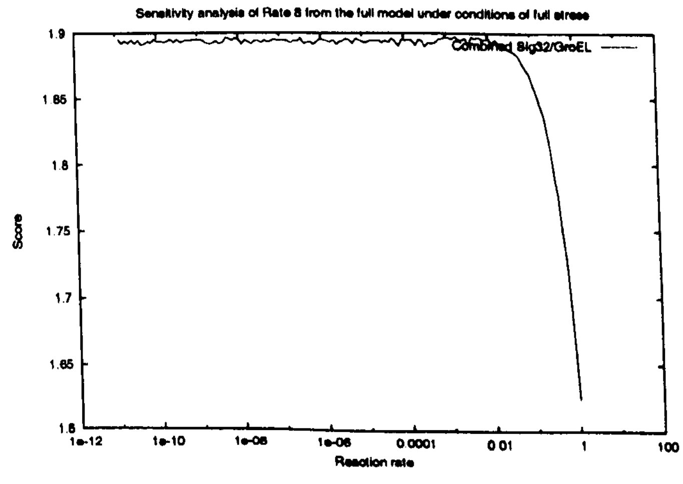


Figure A.8: Sensitivity analysis of Transition 8 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

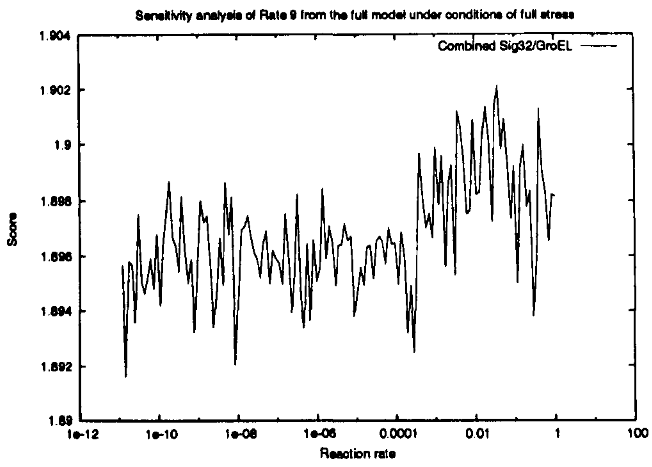


Figure A.9: Sensitivity analysis of Transition 9 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

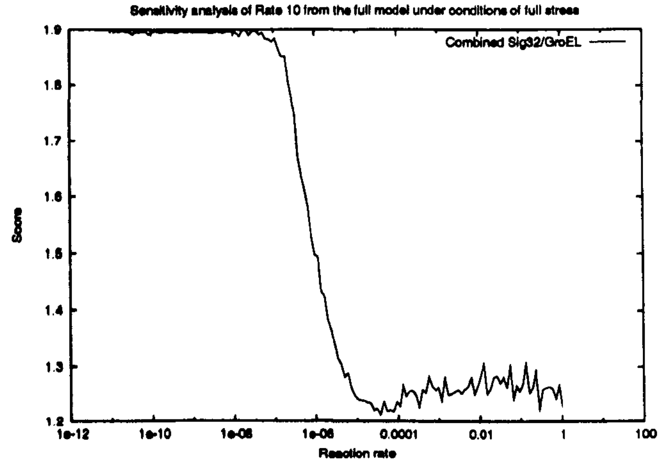


Figure A.10: Sensitivity analysis of Transition 10 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

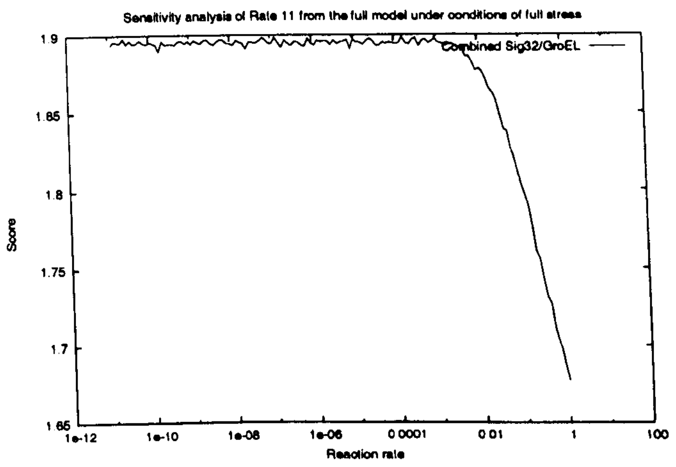


Figure A.11: Sensitivity analysis of Transition 11 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

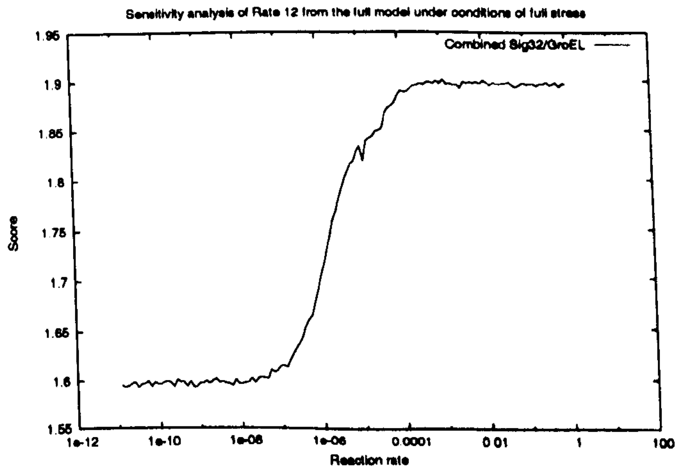


Figure A.12: Sensitivity analysis of Transition 12 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

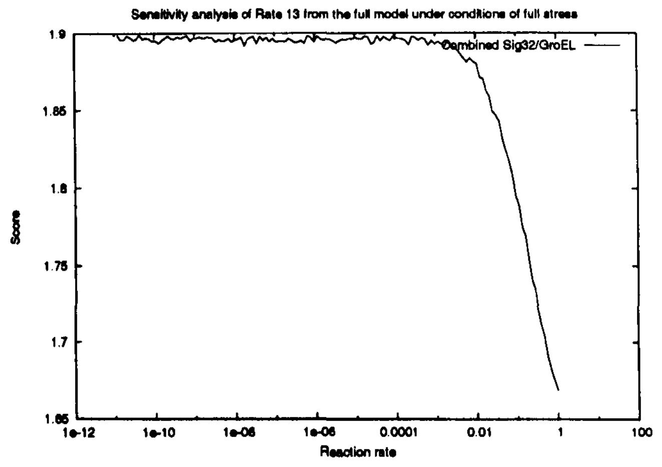


Figure A.13: Sensitivity analysis of Transition 13 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

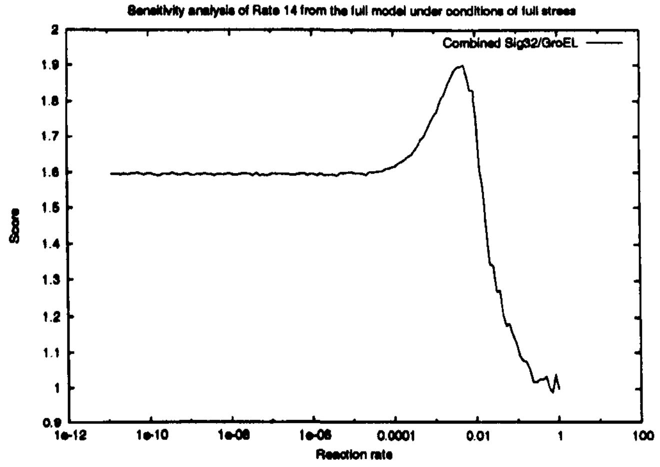


Figure A.14: Sensitivity analysis of Transition 14 from the full model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix B

Sensitivity Analysis of the Full *E. coli* Model under Conditions of No Stress

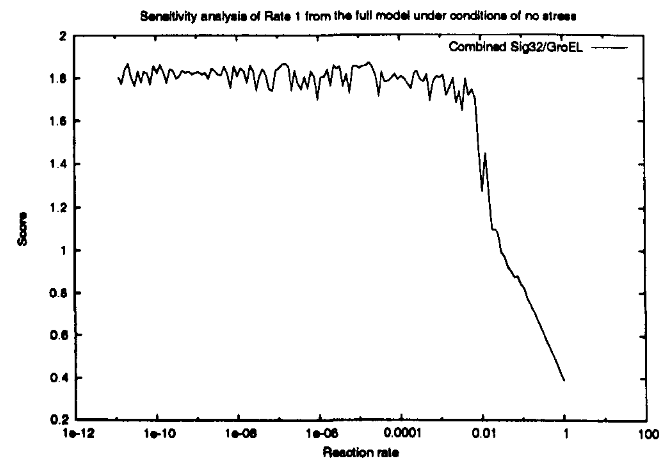


Figure B.1: Sensitivity analysis of Transition 1 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

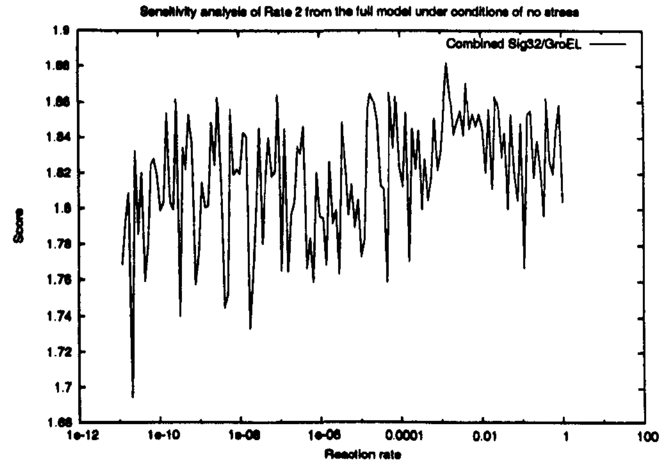


Figure B.2: Sensitivity analysis of Transition 2 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

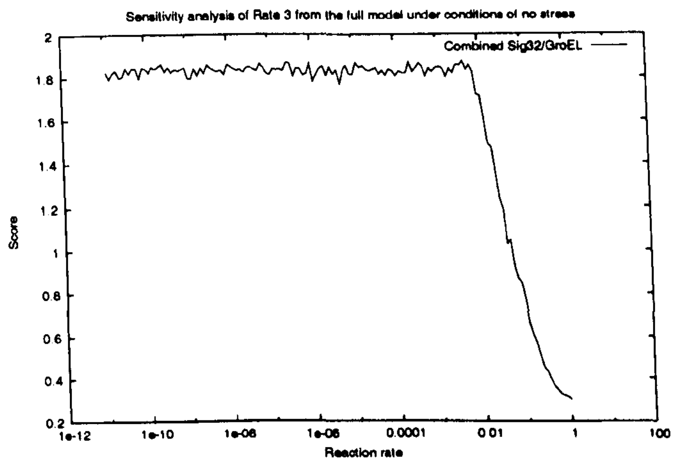


Figure B.3: Sensitivity analysis of Transition 3 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

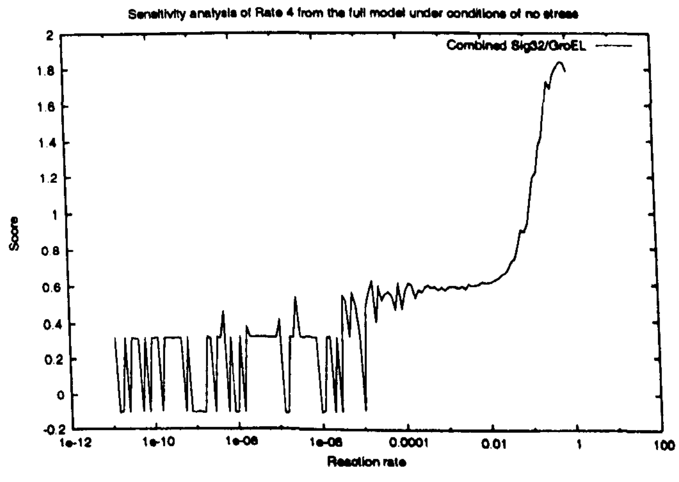


Figure B.4: Sensitivity analysis of Transition 4 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

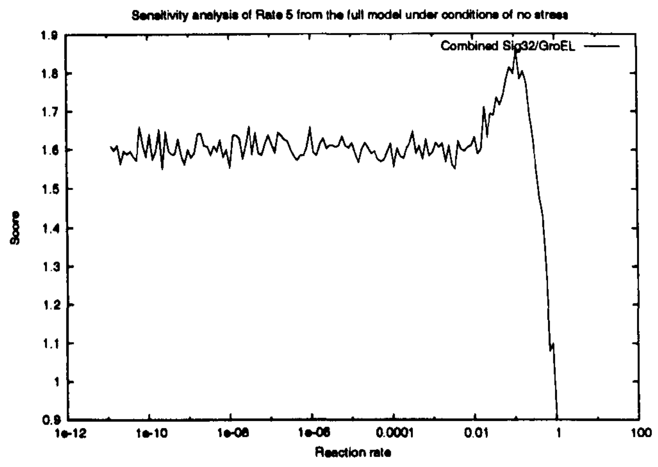


Figure B.5: Sensitivity analysis of Transition 5 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

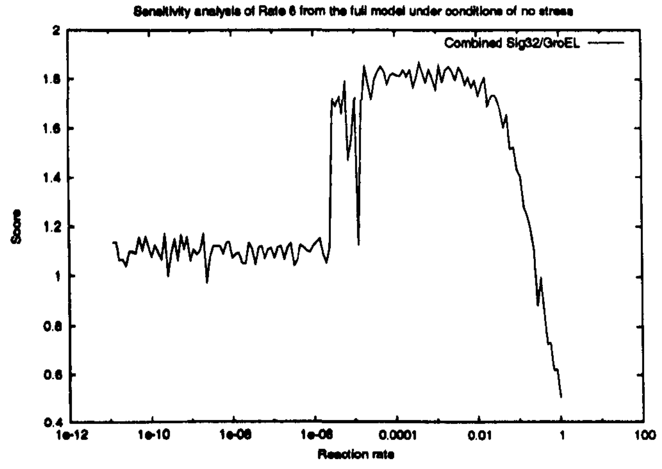


Figure B.6: Sensitivity analysis of Transition 6 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

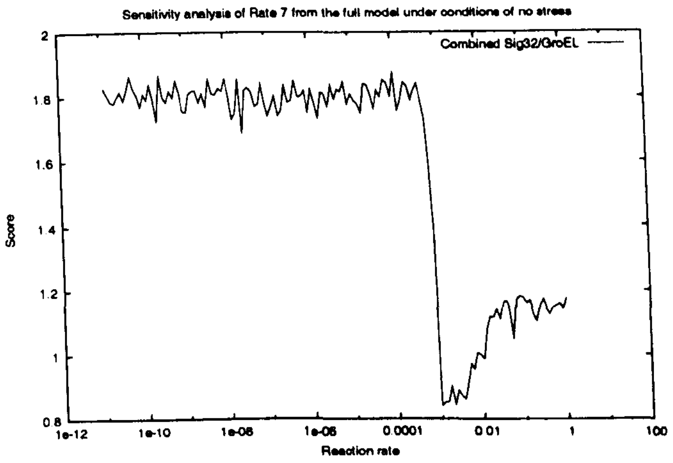


Figure B.7: Sensitivity analysis of Transition 7 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

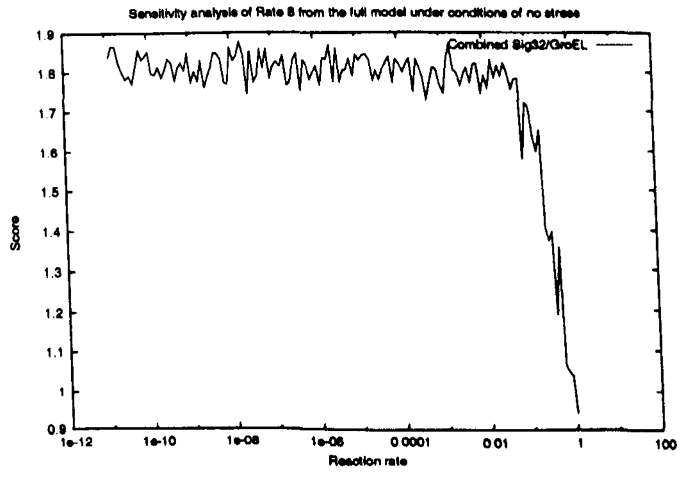


Figure B.8: Sensitivity analysis of Transition 8 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

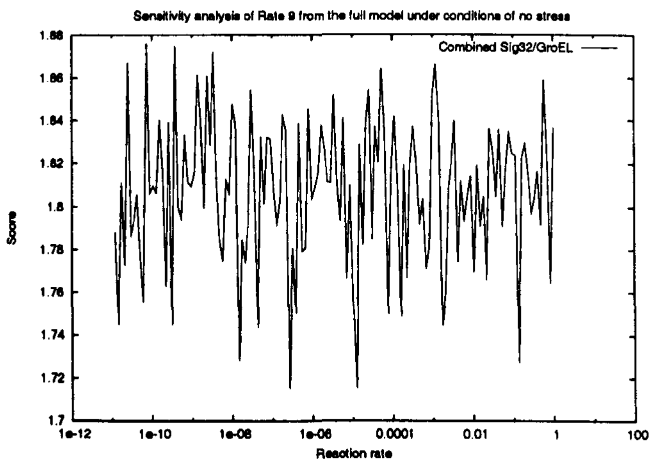


Figure B.9: Sensitivity analysis of Transition 9 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

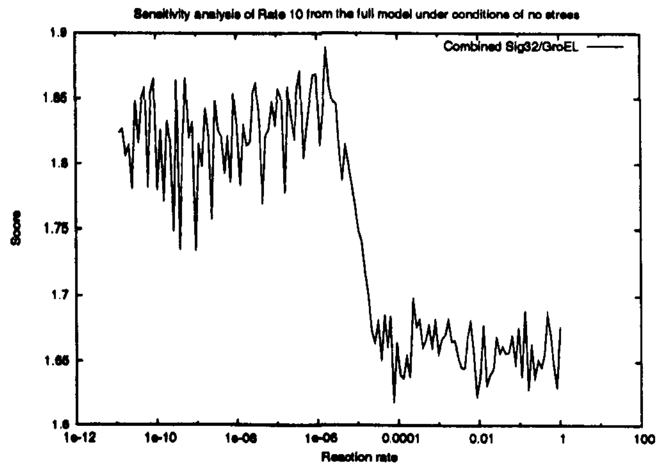


Figure B.10: Sensitivity analysis of Transition 10 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

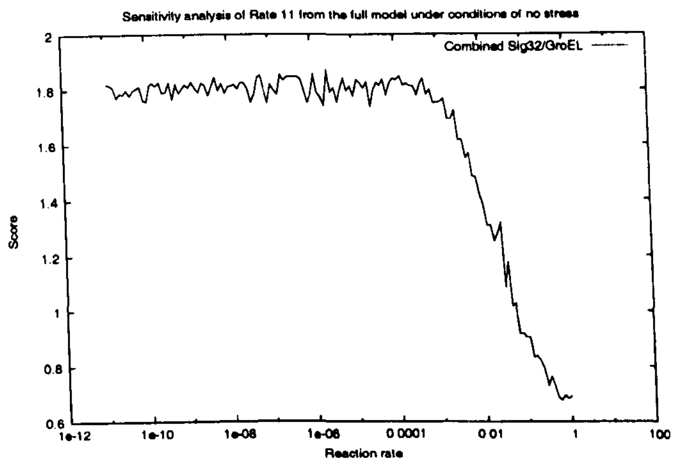


Figure B.11: Sensitivity analysis of Transition 11 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

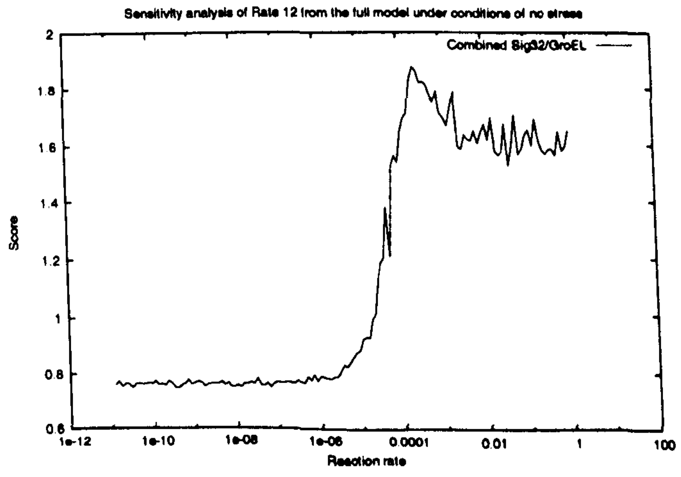


Figure B.12: Sensitivity analysis of Transition 12 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

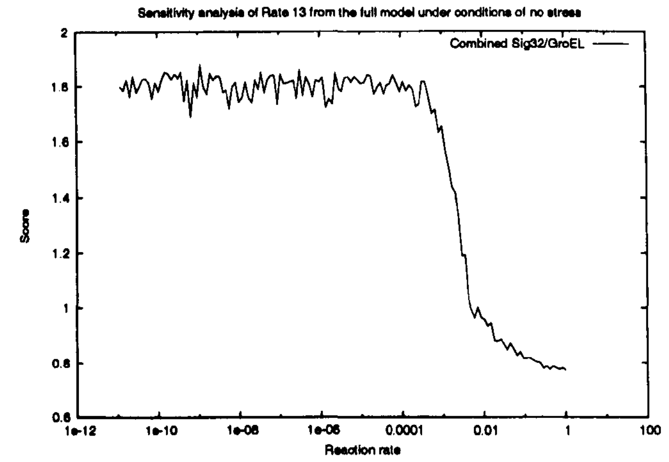


Figure B.13: Sensitivity analysis of Transition 13 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

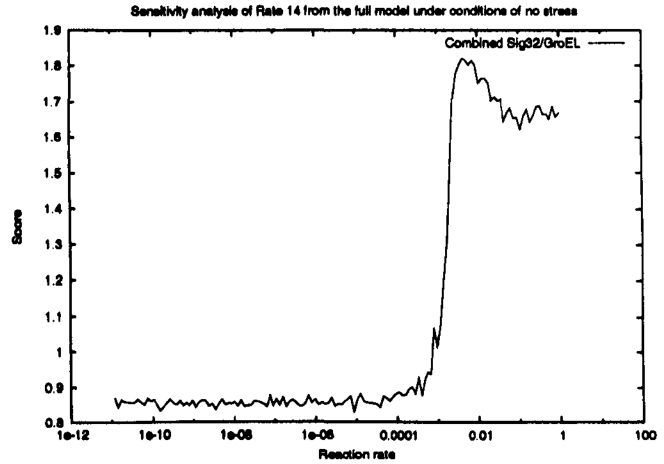


Figure B.14: Sensitivity analysis of Transition 14 from the full model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix C

Sensitivity Analysis of the Full *E. coli* Model Under Anti-sense Mediated Stress

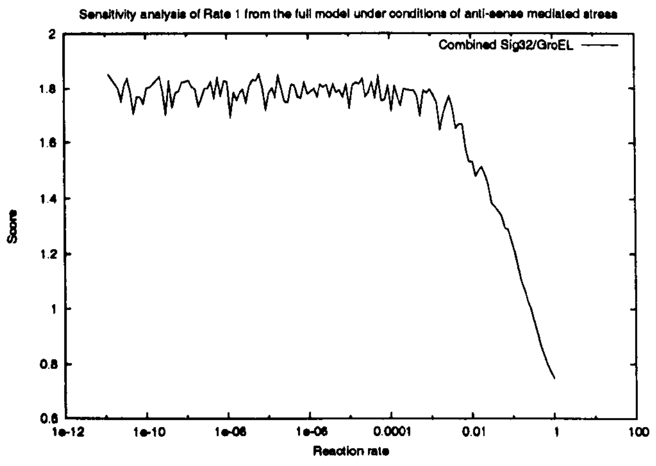


Figure C.1: Sensitivity analysis of Transition 1 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

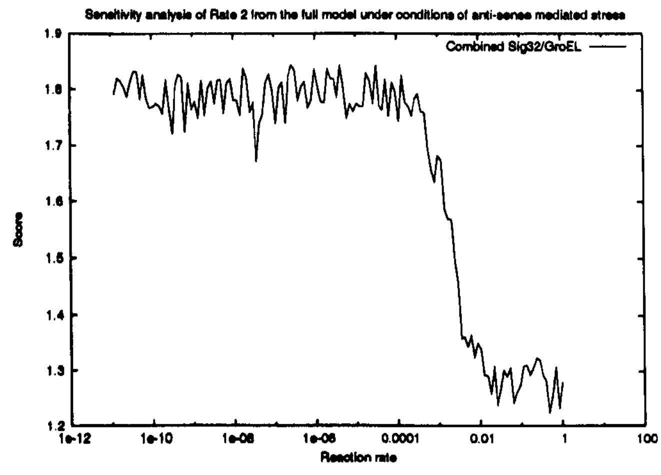


Figure C.2: Sensitivity analysis of Transition 2 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

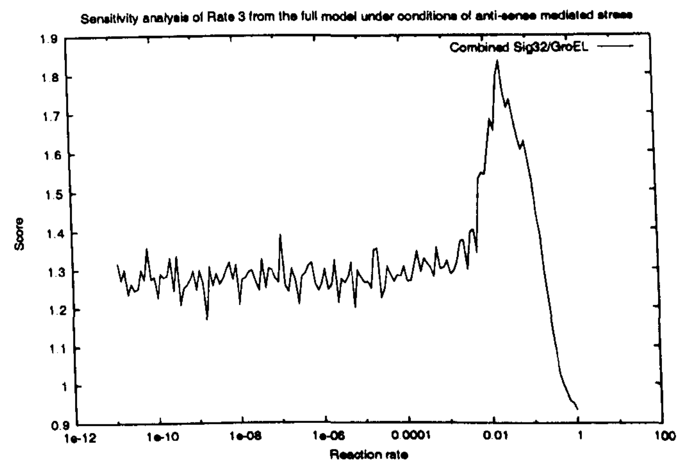


Figure C.3: Sensitivity analysis of Transition 3 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

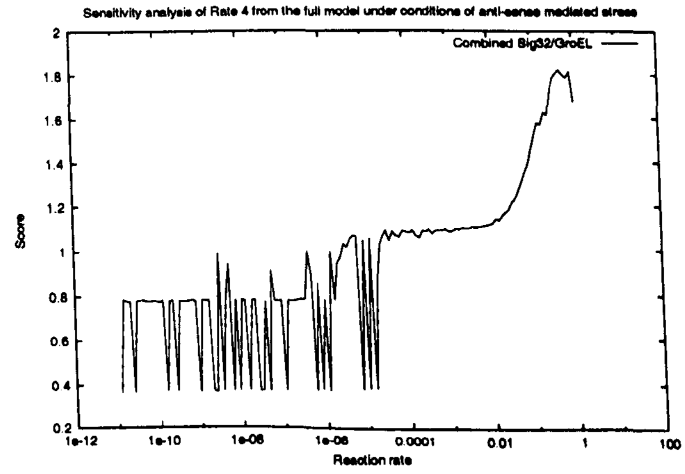


Figure C.4: Sensitivity analysis of Transition 4 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

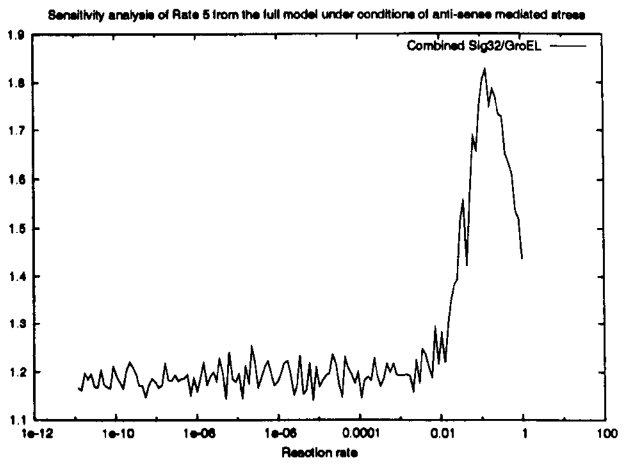


Figure C.5: Sensitivity analysis of Transition 5 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

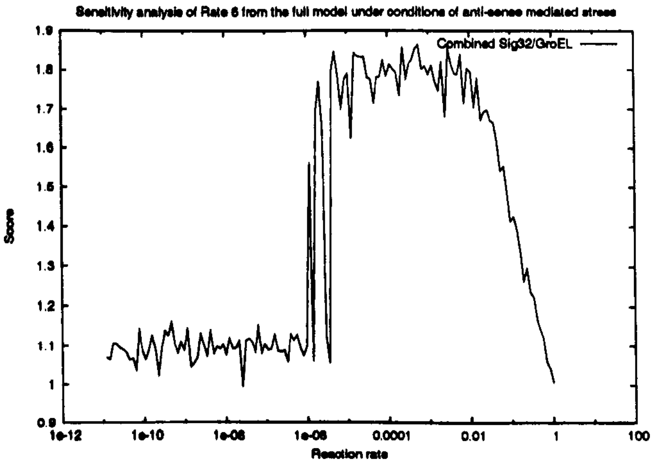


Figure C.6: Sensitivity analysis of Transition 6 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

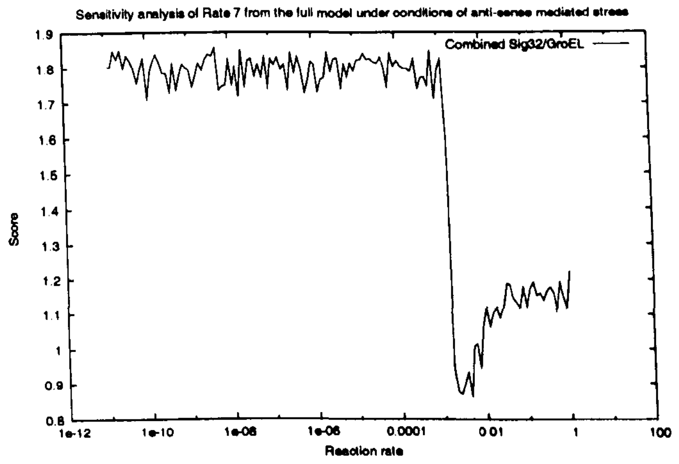


Figure C.7: Sensitivity analysis of Transition 7 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

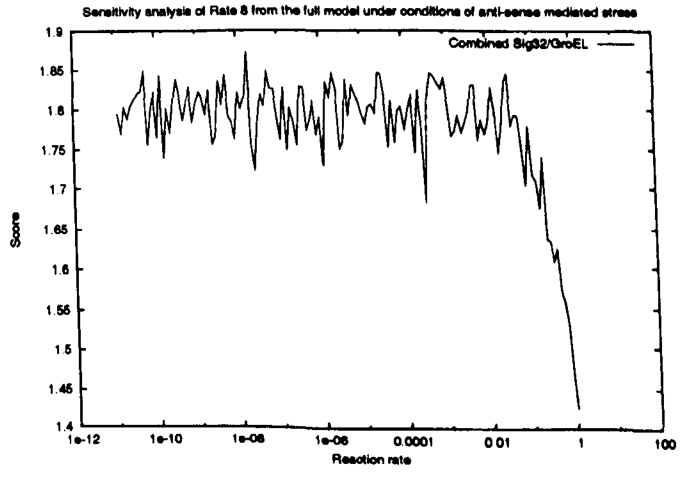


Figure C.8: Sensitivity analysis of Transition 8 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

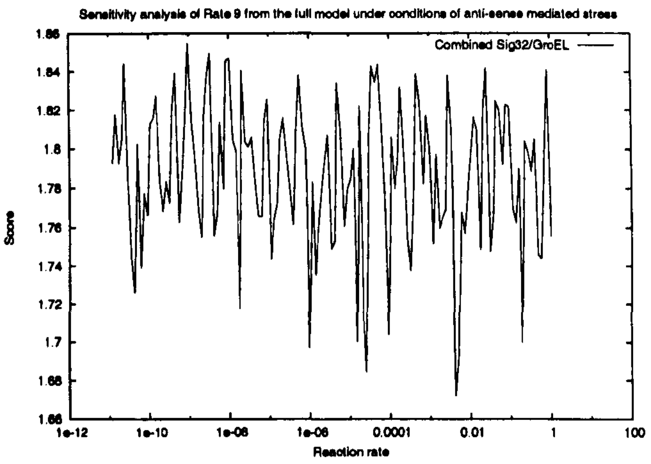


Figure C.9: Sensitivity analysis of Transition 9 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

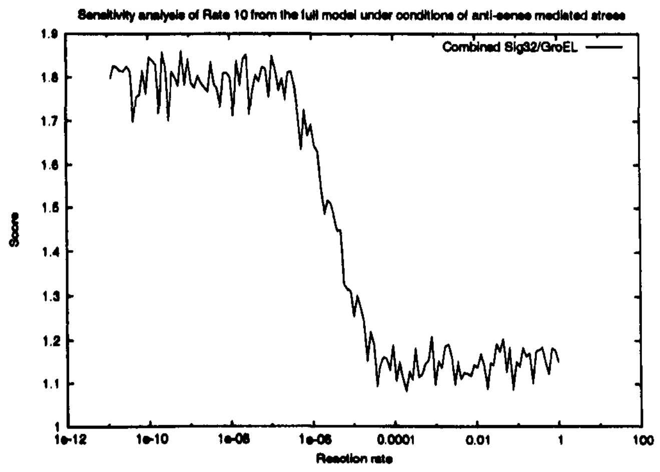


Figure C.10: Sensitivity analysis of Transition 10 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

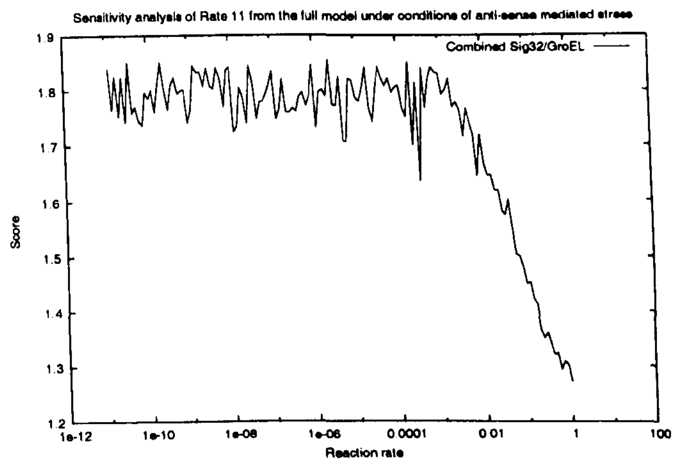


Figure C.11: Sensitivity analysis of Transition 11 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

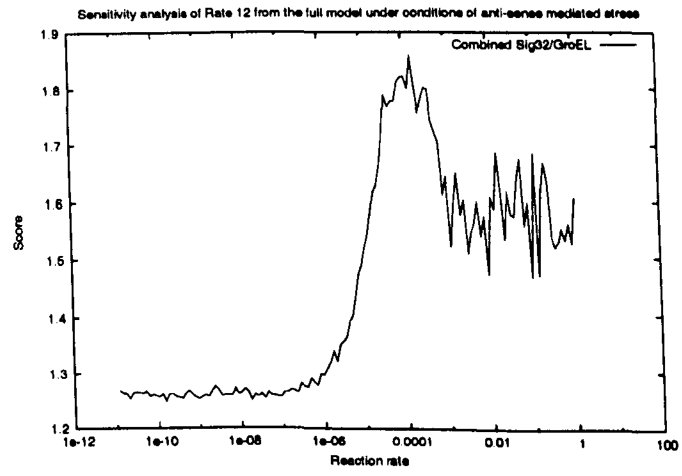


Figure C.12: Sensitivity analysis of Transition 12 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

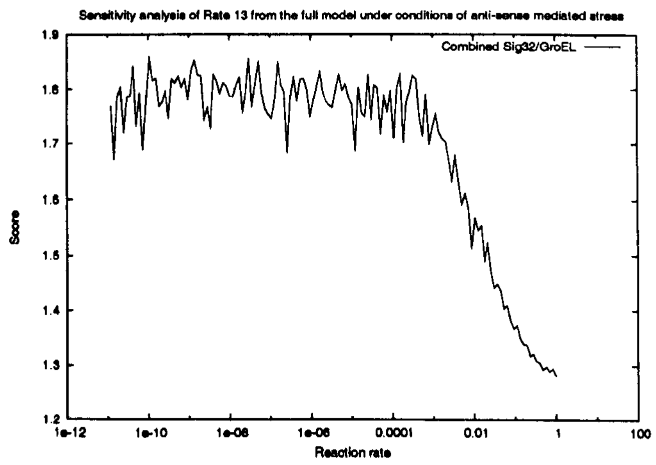


Figure C.13: Sensitivity analysis of Transition 13 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

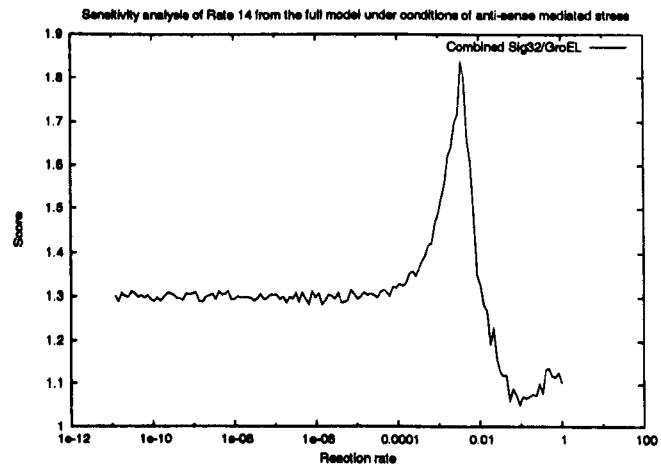


Figure C.14: Sensitivity analysis of Transition 14 from the full model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL/Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix D

Sensitivity Analysis of the Approximate Model of *E. coli* Under Conditions of No Stress

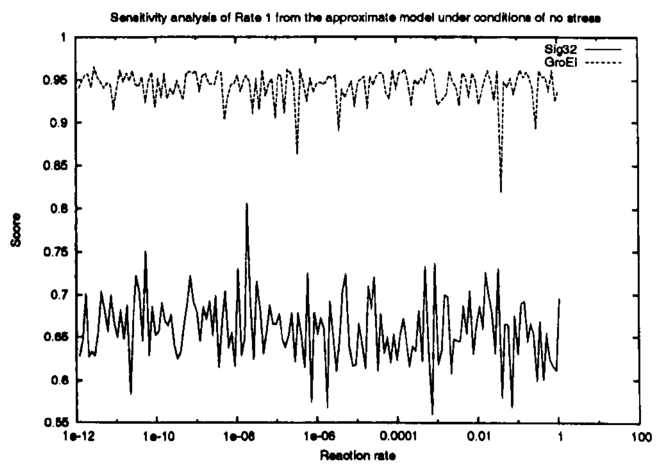


Figure D.1: Sensitivity analysis of Transition 1 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

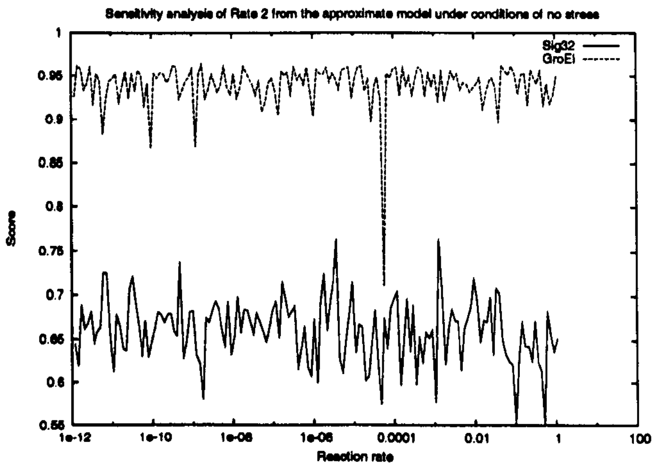


Figure D.2: Sensitivity analysis of Transition 2 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

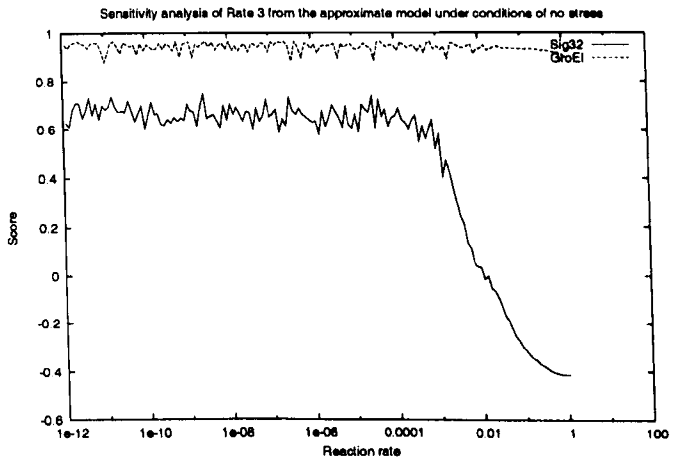


Figure D.3: Sensitivity analysis of Transition 3 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

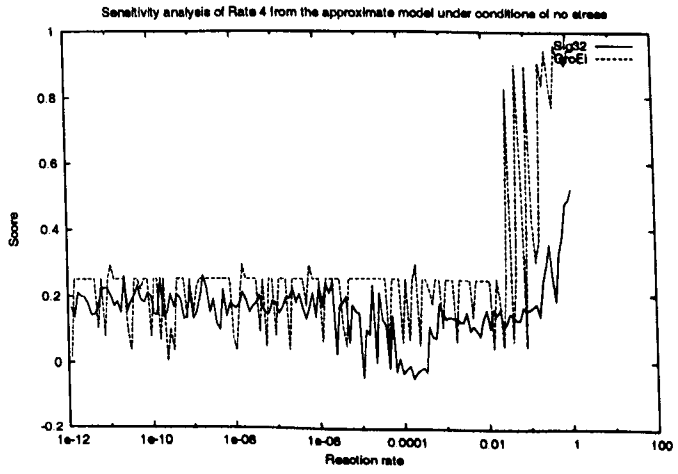


Figure D.4: Sensitivity analysis of Transition 4 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

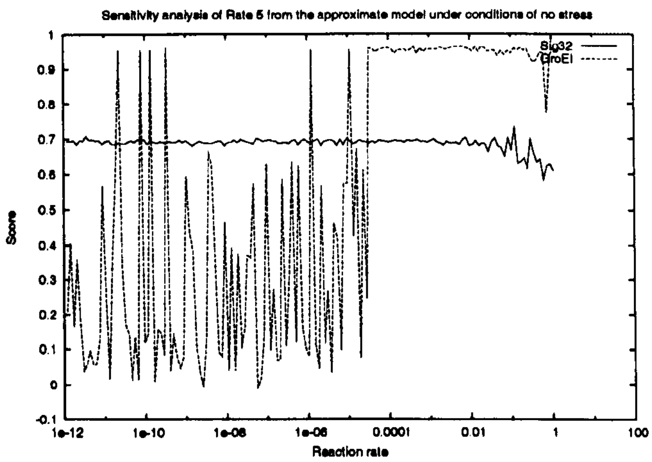


Figure D.5: Sensitivity analysis of Transition 5 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

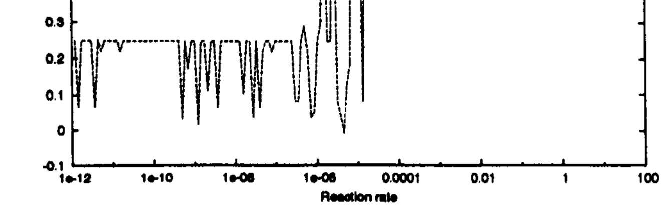


Figure D.6: Sensitivity analysis of Transition 6 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

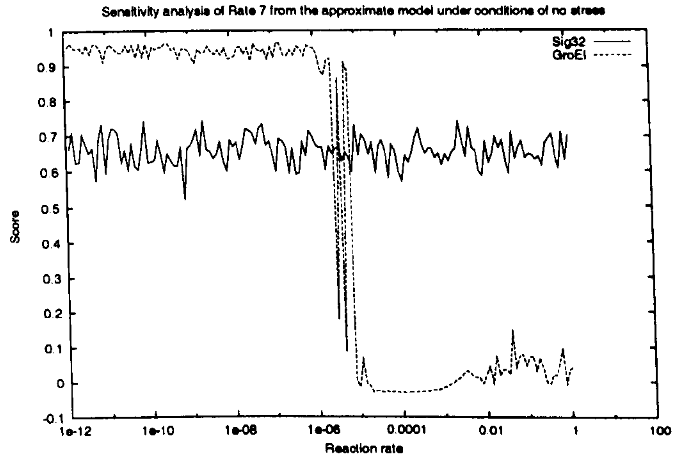


Figure D.7: Sensitivity analysis of Transition 7 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

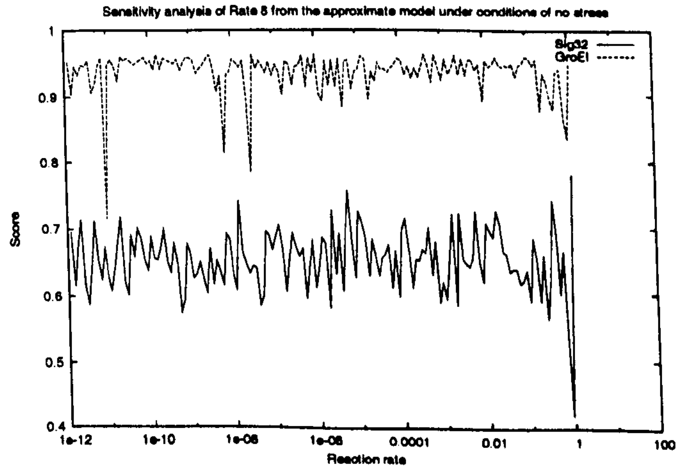


Figure D.8: Sensitivity analysis of Transition 8 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

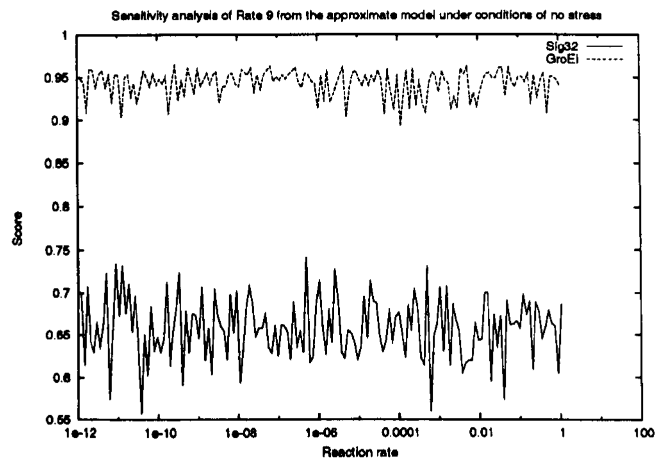


Figure D.9: Sensitivity analysis of Transition 9 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

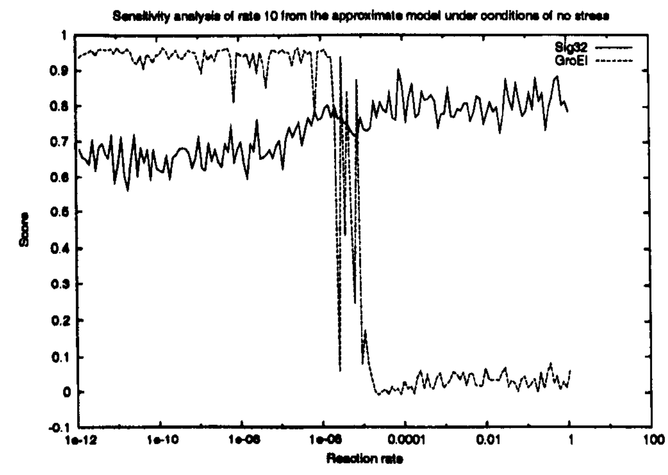


Figure D.10: Sensitivity analysis of Transition 10 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

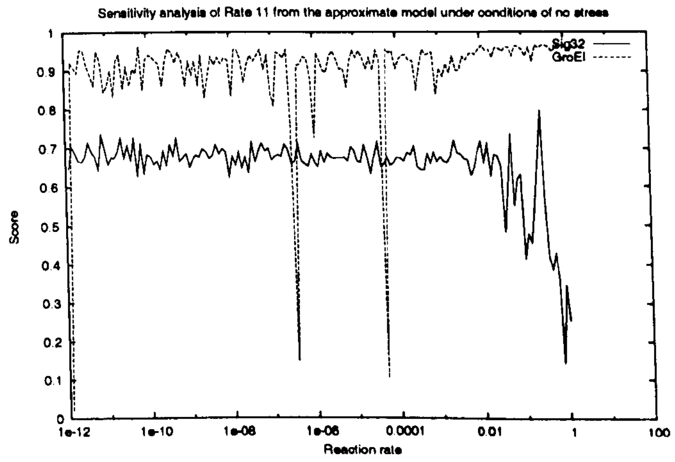


Figure D.11: Sensitivity analysis of Transition 11 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

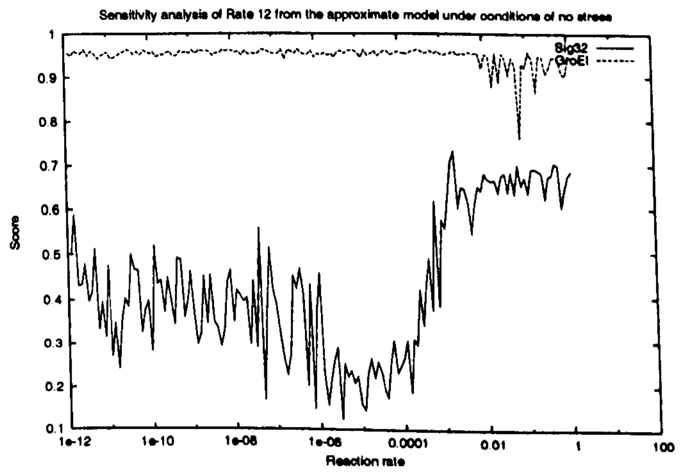


Figure D.12: Sensitivity analysis of Transition 12 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

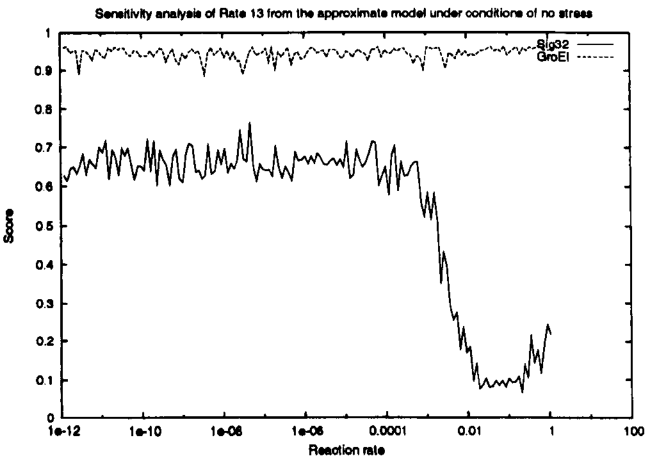


Figure D.13: Sensitivity analysis of Transition 13 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

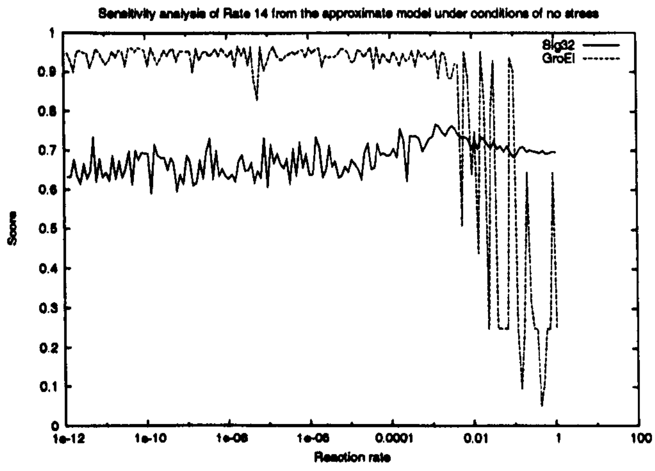


Figure D.14: Sensitivity analysis of Transition 14 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of no stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix E

Sensitivity Analysis of the Approximate Model of *E. coli* Under Conditions of Anti-sense Mediated Stress

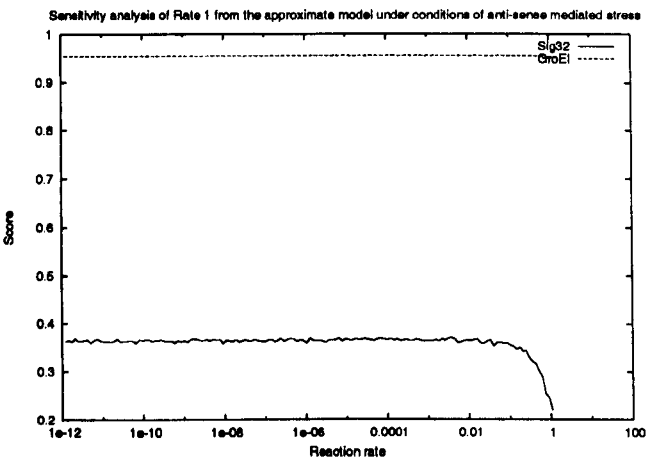


Figure E.1: Sensitivity analysis of Transition 1 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

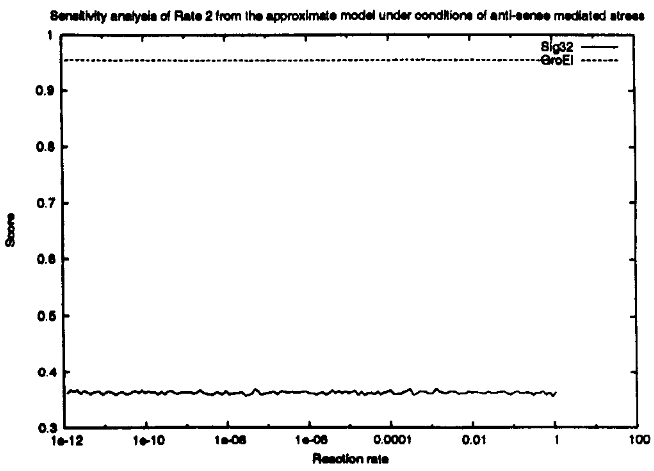


Figure E.2: Sensitivity analysis of Transition 2 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

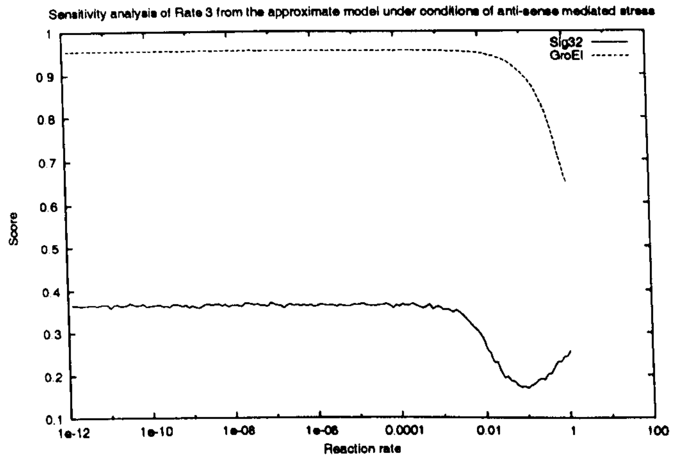


Figure E.3: Sensitivity analysis of Transition 3 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

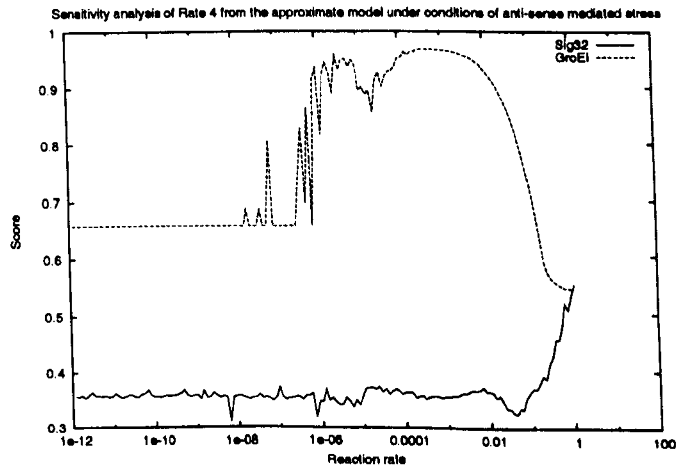


Figure E.4: Sensitivity analysis of Transition 4 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

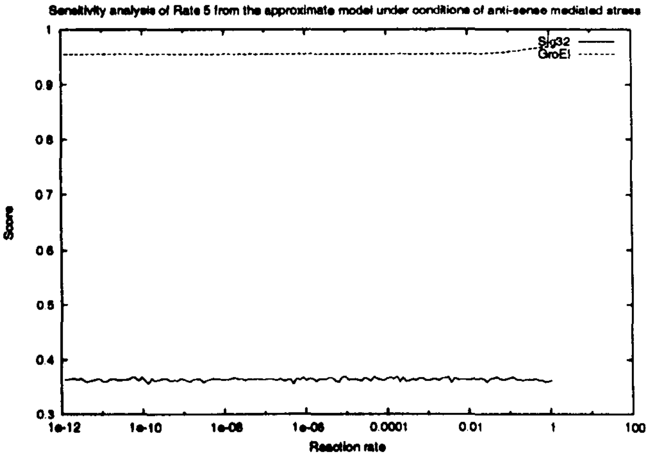


Figure E.5: Sensitivity analysis of Transition 5 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

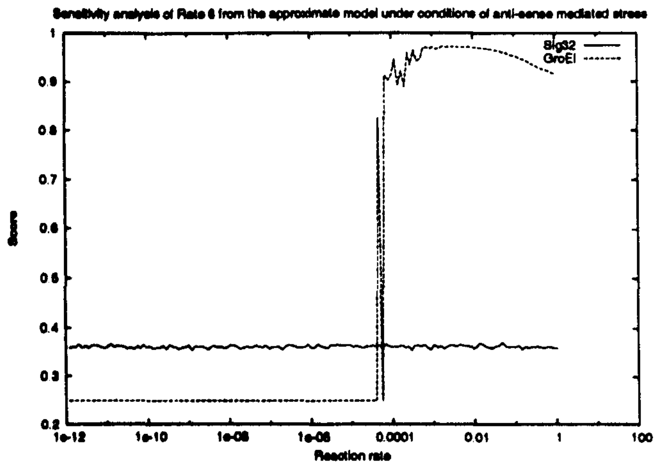


Figure E.6: Sensitivity analysis of Transition 6 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

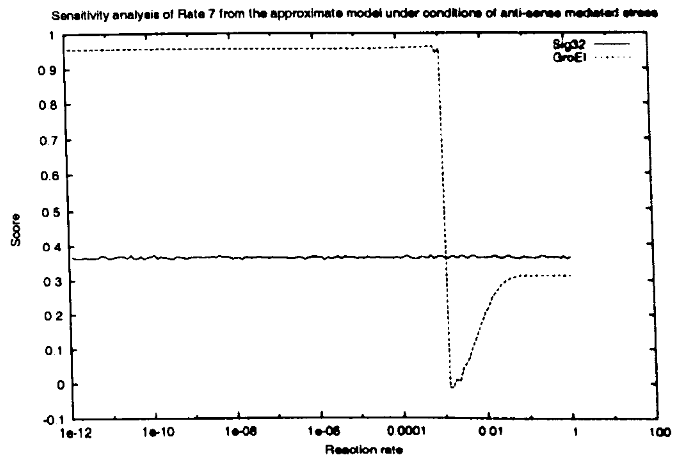


Figure E.7: Sensitivity analysis of Transition 7 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

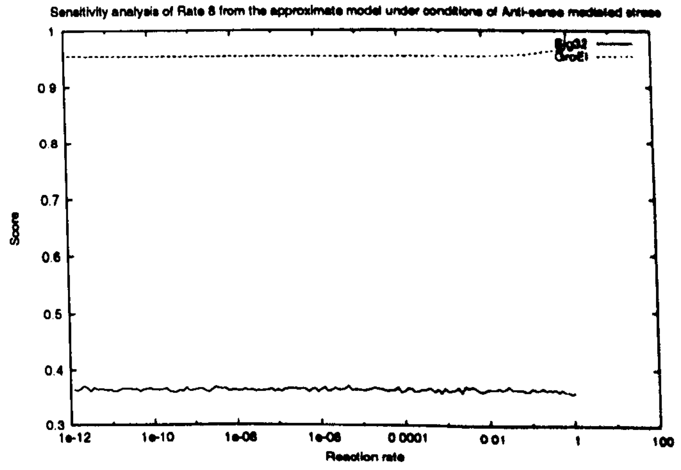


Figure E.8: Sensitivity analysis of Transition 8 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

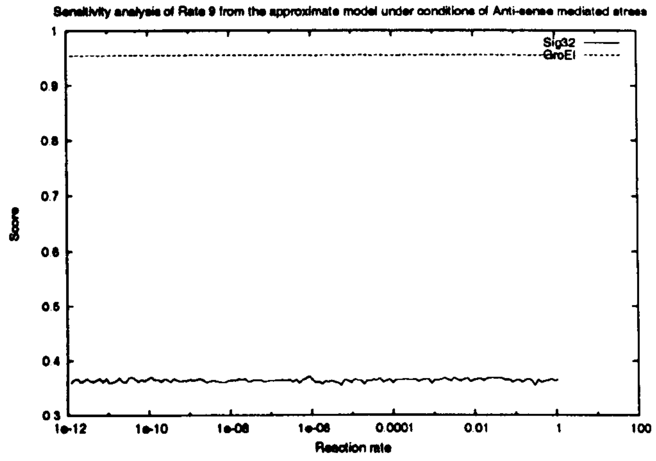


Figure E.9: Sensitivity analysis of Transition 9 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

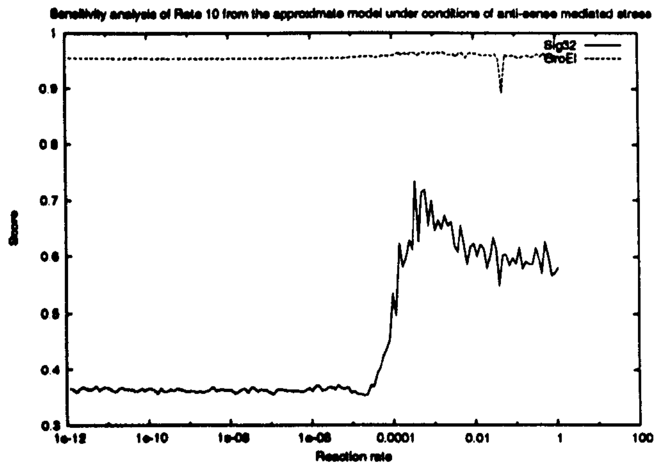


Figure E.10: Sensitivity analysis of Transition 10 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

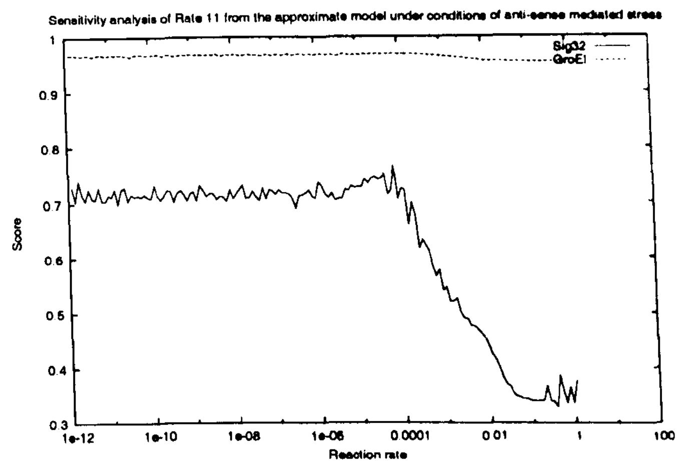


Figure E.11: Sensitivity analysis of Transition 11 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

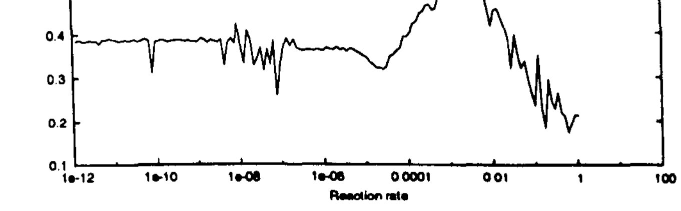


Figure E.12: Sensitivity analysis of Transition 12 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

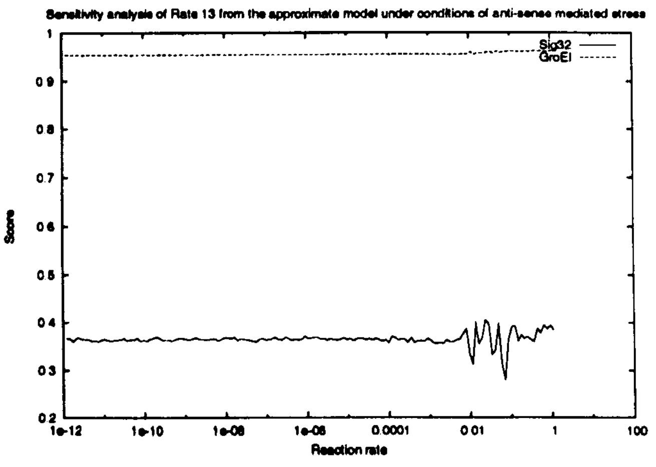


Figure E.13: Sensitivity analysis of Transition 13 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

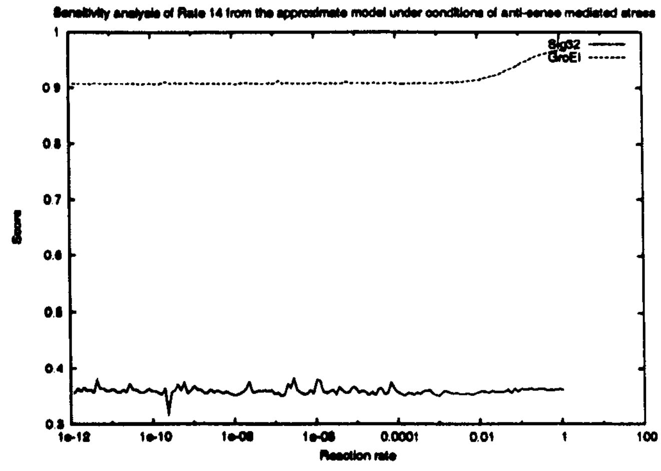


Figure E.14: Sensitivity analysis of Transition 14 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of anti-sense mediated stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix F

Sensitivity Analysis of the Approximate Model of *E. coli* Under Conditions of Full Stress

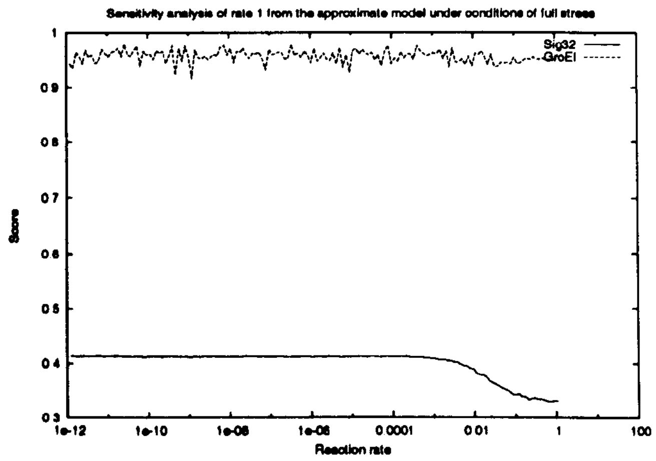


Figure F.1: Sensitivity analysis of Transition 1 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

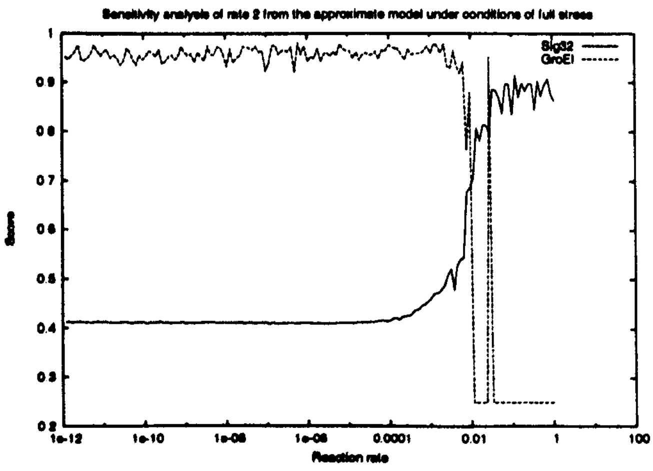


Figure F.2: Sensitivity analysis of Transition 2 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

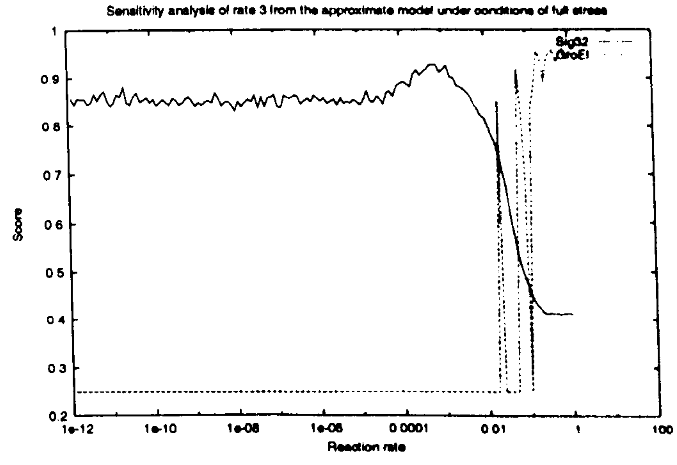


Figure F.3: Sensitivity analysis of Transition 3 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

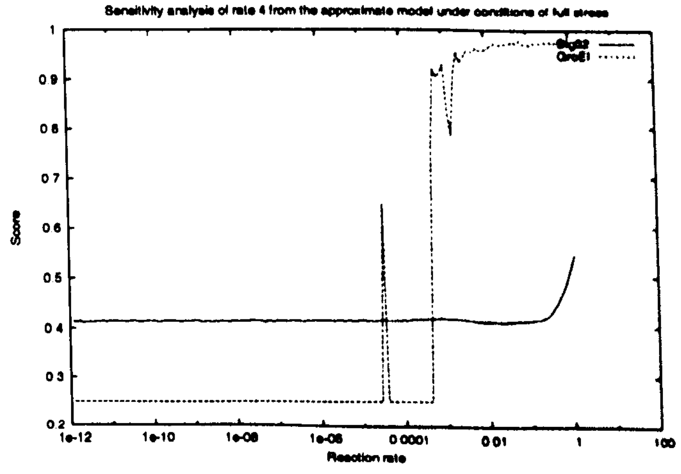


Figure F.4: Sensitivity analysis of Transition 4 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

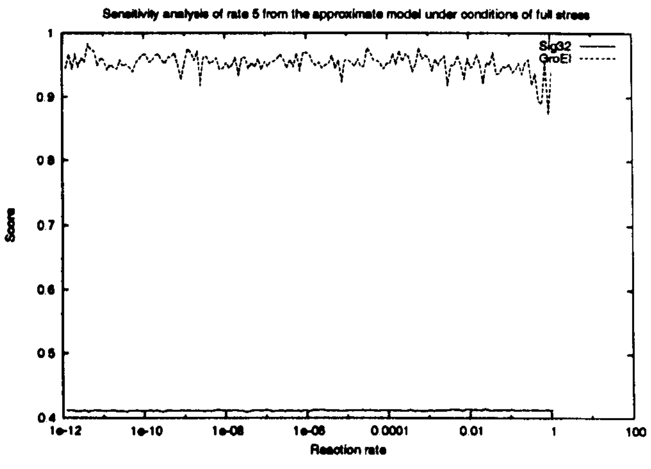


Figure F.5: Sensitivity analysis of Transition 5 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

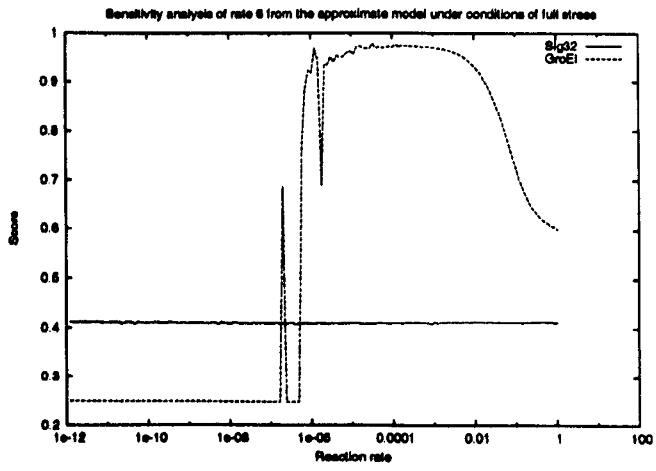


Figure F.6: Sensitivity analysis of Transition 6 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

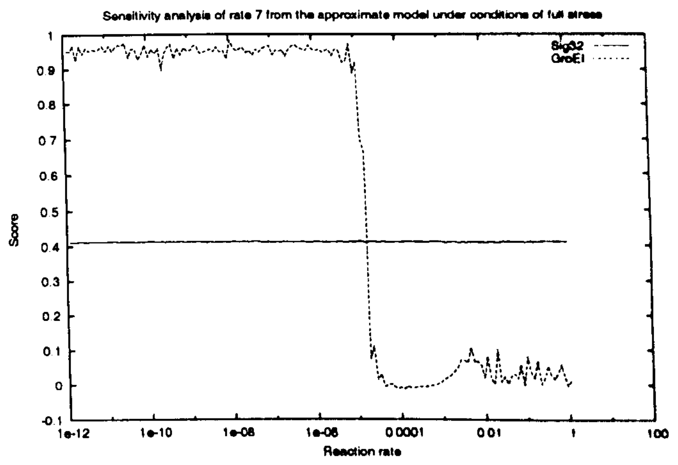


Figure F.7: Sensitivity analysis of Transition 7 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

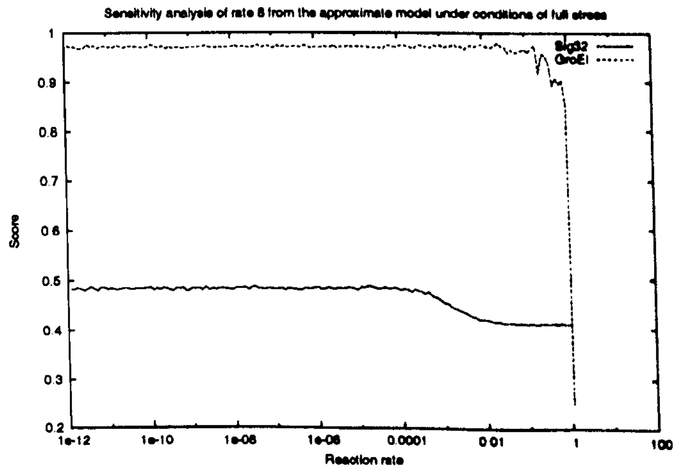


Figure F.8: Sensitivity analysis of Transition 8 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

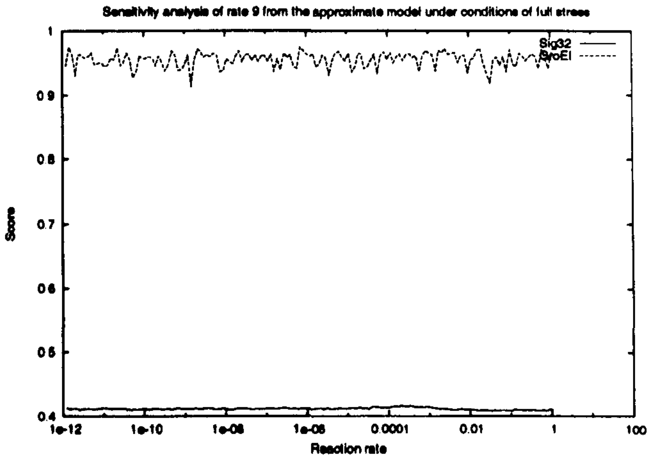


Figure F.9: Sensitivity analysis of Transition 9 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

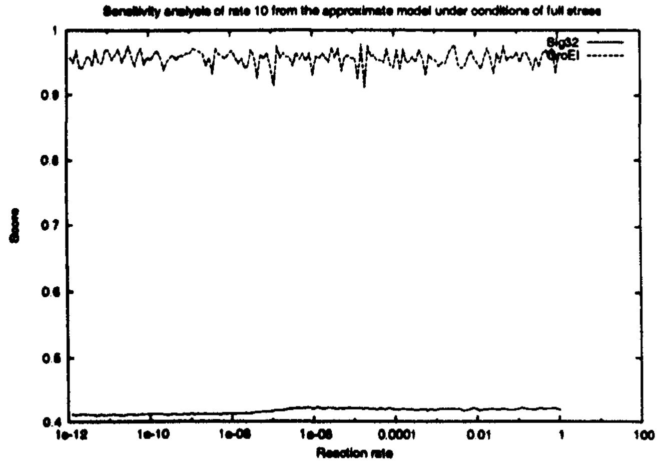


Figure F.10: Sensitivity analysis of Transition 10 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

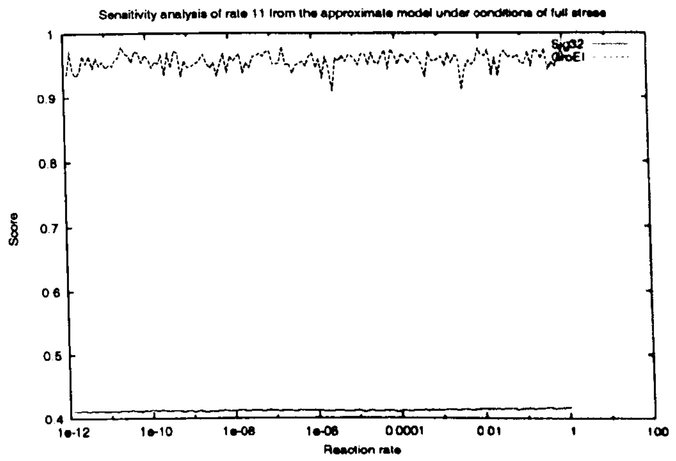


Figure F.11: Sensitivity analysis of Transition 11 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

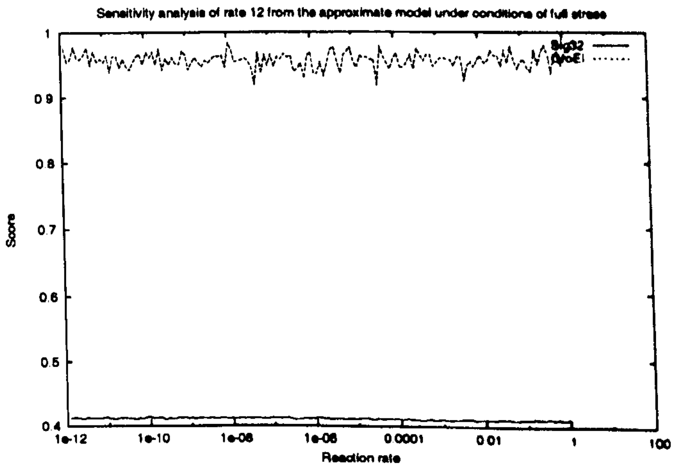


Figure F.12: Sensitivity analysis of Transition 12 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

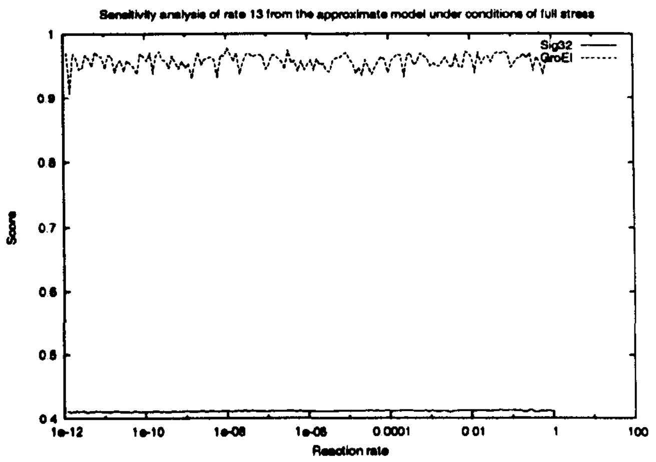


Figure F.13: Sensitivity analysis of Transition 13 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

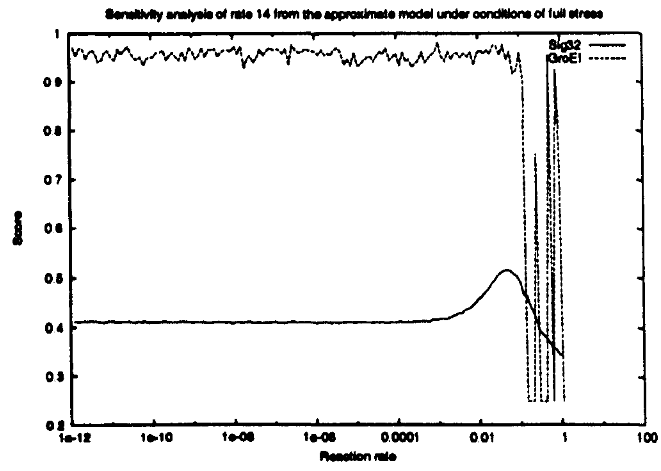


Figure F.14: Sensitivity analysis of Transition 14 from the approximate model of the general stress response of *E. coli* [SPB01] under conditions of full stress. The fitness function score (detailed in Chapter 5) of GroEL and Sig32 is plotted against parameter value, on a logarithmic scale.

Appendix G

Boolean Truth Tables Utilised in Chapter 6

The following truth tables relate to the various Boolean models of various aspects of the phosphate stress response in *B.subtilis*.

G.1 Boolean truth tables relating to the model of the SigB regulon

envStress	act_RsbU
0	0
1	1

Table G.1: The Boolean truth table describing the next state of active RsbU dependant upon the state of its nearest neighbours

RsbV	RsbW_SigB	RsbV_RsbW
0	0	0
0	1	0
1	0	0
1	1	1

Table G.2: The Boolean truth table describing the next state of RsbV_RsbW dependant upon the state of its nearest neighbours

RsbV_P	act_RsbU	act_RsbP	RsbV
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table G.3: The Boolean truth table describing the next state of RsbV dependant upon the state of its nearest neighbours

engStress	act_RsbP
0	0
1	1

Table G.4: The Boolean truth table describing the next state of active RsbP dependant upon the state of its nearest neighbours

RsbV_RsbW	RsbW	SigB	SigB
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table G.5: The Boolean truth table describing the next state of SigB dependant upon the state of its nearest neighbours

RsbV_RsbW	RsbW	SigB	RsbW_SigB
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table G.6: The Boolean truth table describing the next state of RsbW_SigB dependant upon the state of its nearest neighbours

RsbV_RsbW	act_RsbU	act_RsbP	RsbV_P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table G.7: The Boolean truth table describing the next state of phosphorylated RsbV dependant upon the state of its nearest neighbours

SigB	response
1	1
0	0

Table G.8: The Boolean truth table describing the next state of SigB stress response dependant upon the state of its nearest neighbours

G.2 Boolean truth tables relating to the model of the Pho regulon

PhoStress	AbrB	ResD_P	PhoP_P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table G.9: The Boolean truth table describing the next state of phosphorylated PhoP dependant upon the state of its nearest neighbours

Spo0A	PhoP_P	ResD_P
0	0	0
0	1	1
1	0	0
1	1	0

Table G.10: The Boolean truth table describing the next state of phosphorylated RedD dependant upon the state of its nearest neighbours

Spo0A	PhoP_P	AbrB
0	0	0
0	1	1
1	0	0
1	1	0

Table G.11: The Boolean truth table describing the next state of AbrB dependant upon the state of its nearest neighbours

PhoP_P	YkoL
0	0
1	1

Table G.12: The Boolean truth table describing the next state of YkoL dependant upon the state of its nearest neighbours

G.3 Boolean truth tables relating to the model of hyper-induction in the Pho/SigB regulon

SigB	SigA	PhoP_P	YkoL
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table G.13: The Boolean truth table describing the next state of YkoL dependant upon the state of its nearest neighbours

SigB	SigA	PhoP_P	H_YkoL
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table G.14: The Boolean truth table describing the next state of hyper-induced YkoL dependant upon the state of its nearest neighbours

SigB	PhoP_P	YheK
0	0	0
0	1	0
1	0	0
1	1	1

Table G.15: The Boolean truth table describing the next state of YheK dependant upon the state of its nearest neighbours

SigB	PhoP_P	H_YheK
0	0	0
0	1	0
1	0	1
1	1	0

Table G.16: The Boolean truth table describing the next state of hyper-induced YheK dependant upon the state of its nearest neighbours

G.4 Boolean truth tables relating composed model of the phosphate stress response

SigB	SigA	PhoP_P	H_YkoL
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table G.17: The Boolean truth table describing the next state of hyper-induced YkoL dependant upon the state of its nearest neighbours

SigB	PhoP_P	YheK
0	0	0
0	1	0
1	0	0
1	1	1

Table G.18: The Boolean truth table describing the next state of YheK dependant upon the state of its nearest neighbours

SigB	PhoP_P	H_YheK
0	0	0
0	1	0
1	0	1
1	1	0

Table G.19: The Boolean truth table describing the next state of hyper-induced YheK dependant upon the state of its nearest neighbours

envStress	act_RsbU
0	0
1	1

Table G.20: The Boolean truth table describing the next state of active RsbU dependant upon the state of its nearest neighbours

RsbV	RsbW_SigB	RsbV_RsbW
0	0	0
0	1	0
1	0	0
1	1	1

Table G.21: The Boolean truth table describing the next state of RsbV_RsbW dependant upon the state of its nearest neighbours

RsbV_P	act_RsbU	act_RsbP	RsbV
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table G.22: The Boolean truth table describing the next state of RsbV dependant upon the state of its nearest neighbours

engStress	act_RsbP
0	0
1	1

Table G.23: The Boolean truth table describing the next state of active RsbP dependant upon the state of its nearest neighbours

RsbV	RsbW_SigB	SigB
0	0	0
0	1	0
1	0	0
1	1	1

Table G.24: The Boolean truth table describing the next state of SigB dependant upon the state of its nearest neighbours

RsbV_RsbW	RsbW	SigB	RsbW_SigB
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Table G.25: The Boolean truth table describing the next state of RsbW_SigB dependant upon the state of its nearest neighbours

RsbV_RsbW	act_RsbU	act_RsbP	RsbV_P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table G.26: The Boolean truth table describing the next state of phosphorylated RsbV dependant upon the state of its nearest neighbours

SigB	response
1	1
0	0

Table G.27: The Boolean truth table describing the next state of SigB response dependant upon the state of its nearest neighbours

PhoStress	AbrB	ResD_P	PhoP_P
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table G.28: The Boolean truth table describing the next state of phosphorylated PhoP dependant upon the state of its nearest neighbours

Spo0A	PhoP_P	ResD_P
0	0	0
0	1	1
1	0	0
1	1	0

Table G.29: The Boolean truth table describing the next state of phosphorylated ResD dependant upon the state of its nearest neighbours

Spo0A	PhoP_P	AbrB
0	0	0
0	1	1
1	0	0
1	1	0

Table G.30: The Boolean truth table describing the next state of AbrB dependant upon the state of its nearest neighbours

SigB	SigA	PhoP_P	YkoL
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Table G.31: The Boolean truth table describing the next state of YkoL dependant upon the state of its nearest neighbours