Imperial College London
Department of Computing

System modelling of cell signalling pathways

by

Holehouse, A.

Submitted in partial fulfilment of the requirements for the MSc Degree
in Computing Science of Imperial College London

September 2011

Abstract

In this project, we researched, constructed and simulated signalling pathways in the context of glucocorticosteroid resistant asthma. There is significant evidence that p38 MAPK has a role in modulating and disrupting the glucocorticosteroid signalling pathway in corticosteroid resistant asthma patients. By re-constructing a pre-existing p38 MAPK signalling pathway, designing and building a novel glucocorticosteroid signalling pathway and developing a general software tool for integrating two pathways together, we have generated a model whereby with empirical data, an assessment regarding pathway crosstalk can be made. To validate these models, we developed a Monte Carlo parameter estimation tool to generate functional parameter sets, and applied it to both the p38 MAPK pathway and our integrated p38 MAPK - glucocorticosteroid signalling pathway. The p38 MAPK simulations were in line with empirical data, as well as previous simulations done using JACOBIAN and BIO-PEPA. Although we lacked the experimental data to establish the biological correctness of the integrated model, we validated that it behaves in a tractable manner, and represents a stable, functional, multi-branched pathway.

Acknowledgements

I would like to thank Prof. Yike Guo for his support and guidance, Prof. Ian Adcock for valuble discussions regarding the glucocorticosteriod signalling pathway, and Xian Yang for extensive software troubleshooting and general discussion.

I would also like to thank Martha for putting up with my ramblings, Sam Hampton and Lucy Farrimond for everything, and all my other friends, family and coursemates for their unwaivering support and help throughout this project.

Contents

Chapter 1

Introduction

1.1 Overview

Systems biology offers a set of methodologies to simulate and model of biological systems. In this project, we consider the signalling processes which occur in the asthmatic response, and how these may be affected by Glucocorticosteroid (GC), the typical treatment for chronic asthma. Resistance to GC is a major source of asthmatic complications, and accounts for a significant proportion of both the mortality rate and the cost associated with the disease. By investigating these signalling pathways, a better understanding of the disease's molecular mechanism, and more specifically the resistance mechanism is envisaged. Based on these developments, more effective research targets, drugs and treatments for these Corticosteroid Resistant (CSR) patients may be possible.

To achieve this goal, we constructed two models of signalling pathways, developed a piece of software to (generally) integrate signalling pathway models together in a semantically and syntactically correct manner, and then carried out simulations on both one of the induvidual pathways, and the integrated system. By comparing the results of these simulations to both previous work and one another, the model's basic validity has been proven. We open the door to interdiciplinary research between pharmacologists and systems biologists to dynamically explore this new integrated pathways properties, with a view to identifying signalling crosstalk between the two arms.

Initially, background information regarding the state of the art and a brief biological refresher will be presented, with methods of model simulation discussed and our choice of ODEs using MATLAB justified. We consider some of the features associated with Monte Carlo parameter estimation, and include an overview and description of the Systems Biology Markup Language (SBML) specification. Next, a discussion on asthma, GC resistance, and the challenges facing medical practitioners and researchers alike is presented. An overview of the work which directly preceded this project is done, and a justification of our approach is presented. With this background in place, it becomes apparent how the project has been divided up.

Chapter 2 outlines the design process and implementation of SBMLIntegrator, a stand alone command line Linux based software for integrating two SBML models together. We consider the semantic challenges associated with model integration, some of the functional approaches used in our software architecture, and the development style and tools used to complete the work. The source code is provided, along with complete code documentation, installation guide and a manual for use in the supporting information. In addition to its current state, we discuss future development.

Chapter 3 describes the work involved in taking a pre-existing SBML model of the p38 MAPK pathway, de-constructing it to ensure it meets the SBML standard, and reconstructing it in a manner whereby MATLAB based ODE simulations can be performed using it as a base model. Beyond this, using the design concepts upon which this original model was built and extensive background

research, we discuss the novel design of a model of the GC signalling pathway. Finally, we discuss the process of integrating these two pathways together using SBMLIntegrator.

Chapter 4 shows the work done with both our reconstructed p38 MAPK model and the new integrated model, and describes the design process for a basic Monte Carlo parameter estimation algorithm, implemented in MATLAB. This tool randomly generates a parameter set of rate constants and initial concentrations, simulates the system with these parameters, and evaluates the success of the resulting output by comparing the simulation data with real experimental data.

Chapter 5 provides a cursory analysis of the data generated by our simulations, confirming the validity of both of our parameter estimation approach and the integrated model. We look at some of the features of the generated data, and highlight a small number of biologically relevant hallmarks of the data.

Chapter 6 offers a conclusion the project, and considers future work and development, highlighting not only potential work, but additionally genuine paths of development being explored at this moment in time.

Below we include a summary roadmap detailing the project's progress along an approximate time scale, describing how the discrete components of the project combine together.

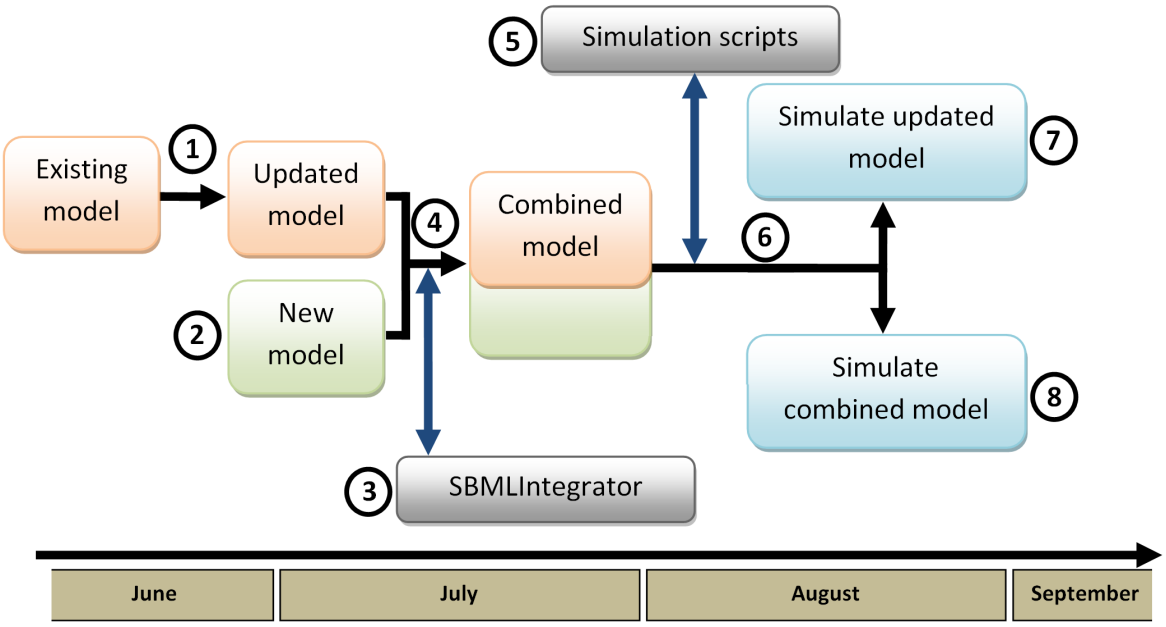

Figure 1.1: Overview roadmap describing the project's timeframe. **(1)** An existing model (p38 MAPK) was taken, updated, and reformatted. **(2)** An entirly new, complementary model was designed and built. **(3)** Realising the challenges associated with manually integrating two models, a software tool to automatically integrated two models together was developed. **(4)** Using that tool the two seperate models were combined into a single integrated model. **(5)** A set of MATLAB scripts were produced which ran both the induvidual model and the integrated one in a simulation. **(7)** We first evaluated the results of the induvidual model to confirm the systems validity. **(8)** Finally, the results of the integrated model simulations were analysed, confirming its overall validity.

1.2 Systems Biology

Systems biology is the application of multiple sets of techniques from a wide range of disciplines to develop, improve and re-define our understanding of the natural world, providing theoretical models to describe natural occurrences. Initial work on membrane action potential by Hodgkin and Huxley [33] led to one of the earliest and most widely used biologically relevant mathematical models (the Hodgkin-Huxley model). Since then, various fields and techniques providing a computational means for biological simulation and modelling have been developed to great effect, providing insight and information which would be impossible to obtain under conventional means. These techniques include those such as Metabolic Control Analysis (MCA)[39][22], agent based modelling[67], network analysis, [40] and molecular dynamics simulation[60], each of which provides unique information relating to the system being described, although their applications differ significantly.

Ultimately, however, these techniques are simply different approaches to the same task - generating a model of a biological system. A model, in this context, can be seen as an abstracted collection of parameters and constraints, which aim to describe the features (static and dynamic) of the source representation. Typically (although not necessarily) models provide a simplification, giving us a tractable set of information, from which we can determine information which may not be readily available from the individual components of that model. The techniques described above differ primarily in the mechanism by which they simplify and aggregate biological information, so as to allow simulation of a hugely complex system using the hardware available today. As a consequence of the aggregation techniques used, different aspects, details and dimensions of a system are lost, such as molecular detail, an extensive time scale, or a system of a meaningful size.

The complexity within biology should not be underestimated. A single eukaryotic (e.g. mammalian) cell contains thousands of proteins [19], many more strands of Ribonucleic Acid (RNA) and Deoxyribonucleic Acid (DNA), and a wealth of other small molecules such as lipids, phosphate based compounds and ions. These biochemical species interact together in countless combinations depending on a huge rage of factors, ranging from those triggered by extracellular events to more predicable activities relating to the properties of the different components. Like a city, different biochemical species, or combinations of species, have a huge range of roles to play, such as cellular maintenance, metabolic anabolism and catabolism, information processing, replication, and cell death. Different species play different roles depending on when in the cells life cycle they appear and exist, where in the cell and its associated substructures they are localized, and what other species are present or not present while they act. This culminates and a near inconceivable level of complexity in every cell, and with the human body containing between 30-100 trillion cells (3×10^{13} to 1×10^{14})[29], with a huge range of different cell types and environments, the complexity facing biochemists begins to become apparent [66][38]. Entwined in this complexity, however, is a significant opportunity for new branches of research, as well as an explosive increase in the quality provided by traditional systems biology techniques

This is a golden time for computational biology, and by extension, for biological research as a whole. In the last fifty years the development of the computer has transformed quantitative analysis from laborious, long winded work to the ubiquitous mainstay of every business, research project and development in virtually every sector of the modern world. Simultaneously, in part through the development of advanced electronics and through an increasingly sophisticated set of tools at experimentalists' disposal, more and more data regarding many different aspects of biology is being uncovered. The combination of a drastic reduction in cost relating to much of this data acquisition, combined with the exponentially growing power of processors means huge amounts of data are being generated which we are only now approaching a stage where we can effectively process, analyse and interpret the data being generated. The complexity, and the number of events occurring in a cell is vast, but largely static, while the number of calculations which can be performed a second is constantly growing. It is not unreasonable to assume that in the next twenty

to fifty years, unanticipated advances in science and technology will yield increasingly powerful electronics and software. With advanced methods and hardware, there is no reason why in the future, systems approaching in size to a cell could not be simulated in increasingly fine detail. While systems biology has previously been seen as the blunt end of research, where guestimation meets ill-fitting parameters and too many assumptions are made to create a biologically relevant models, as both the computational power available and our understanding of the systems increase, more and more of these subtitles can be described. No one is suggesting that systems biology (or indeed the broader scope of computational biology) should replace wet-lab based research. Instead, however, it can provide an additional angle and guidance tool, suggesting possible avenues for researchers to explore, avoiding expensive and time consuming research which may never work.

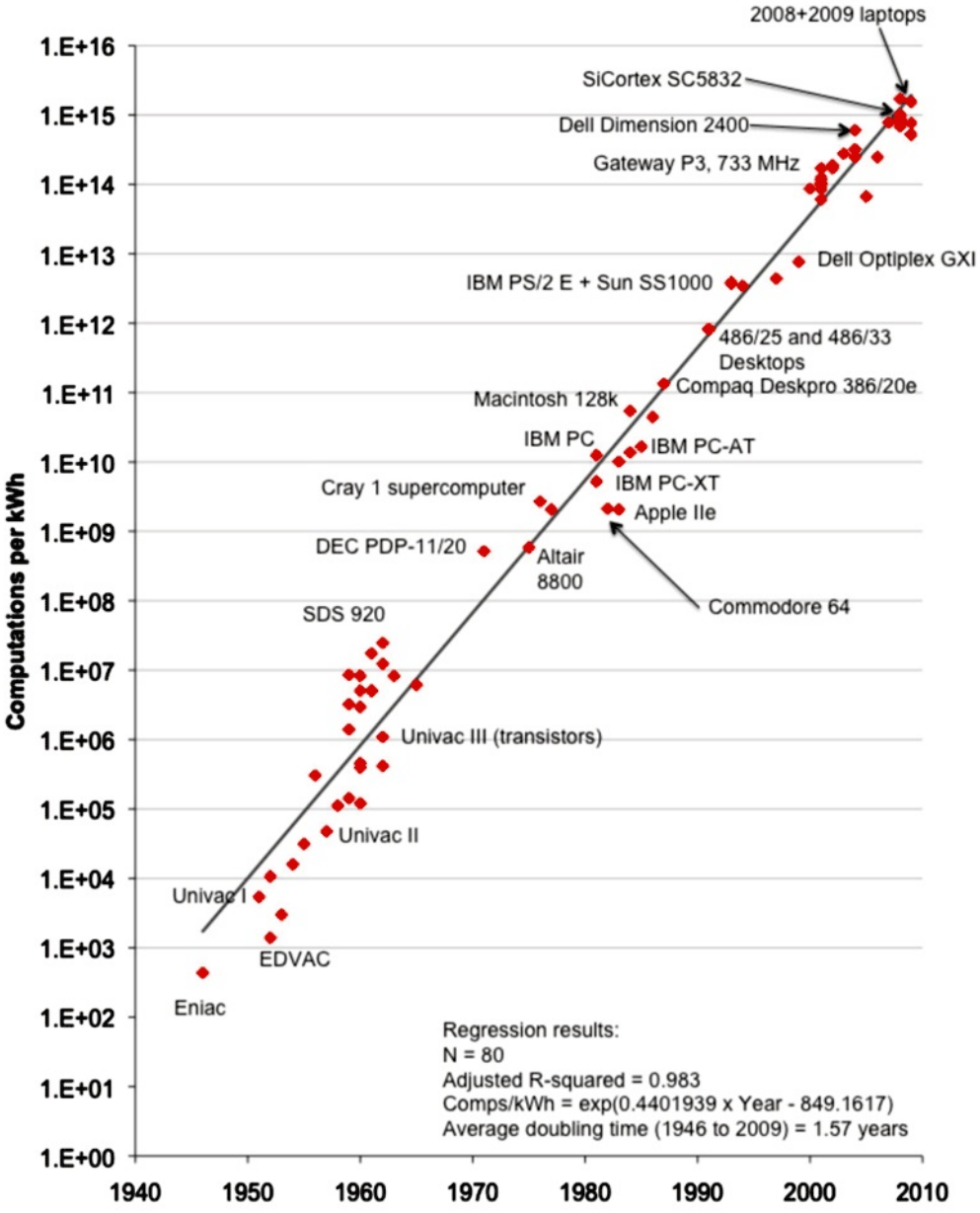

Figure 1.2: Graph describing number of computations per kWh over time (©Jonathan Koomey 2011[43])

1.3 Biochemistry Overview

To effectively describe a biological system, there is a need to carefully define the constraints and parameters involved in that system. To understand the context of these factors, it is advisable to have at least a general understanding of the key components of biological systems. The infinite and repetitive nature of life is primarily driven by the replicative nature of cells. These act as homoeostatic micro environments to facilitate ideal conditions for a wide array of biological functions. Cells come together to form multicellular organisms, where different cells have different roles. Additional components made of protein and/or inorganic material can be exported by the cells to construct huge, complex structures to provide a framework for more complex life. This is how the human body is built - a mass of cells draped around a bone scaffolding, held together by proteinaceous connective tissue.

Information flow and cellular replication are intertwined by the central dogma of molecular biology. DNA provides a long-term storage molecule for information, which can be transcribed into Messenger Ribonucleic Acid (mRNA). mRNA acts as a malleable and short term information transfer molecule, and is used as a blueprint by the cells molecular machinery to assemble proteins. A macromolecular polymer of amino acids, proteins come in a huge range of sizes and roles, from the relative simple structural protein collagen, a primary component of connective tissue [63] to the vastly complicated and multifaceted apoptosome complex, responsible for the coordination and control and apoptosis - programmed cell death[1]. Through the manufacturing of different proteins, cells can construct machinery to carry out all the activities they need to perform. Evolutionary pressures have driven the survival of cells whose DNA produces the proteins best suited to carry out the tasks which lead to their survival in the local environment.

1.3.1 Protein Expression

Protein expression is the process of taking a segment of DNA, transcribing the information encoded in that DNA and translating it into a protein. This is a very complicated process, especially in eukaryotes, but for the purposes of this dissertation a short overview of the critical steps is relevant. Ligand[1] (transcription factor) triggered DNA expression typically occurs through a number of steps. Initially, the transcription factor binds to a DNA promoter region in a reversible fashion. Once this initial binding has occurred, other transcription factors associated with the DNA-ligand complex interact to form the transcription apparatus. With the transcription apparatus in place, it proceeds linearly along the DNA, using individual RNA nucleotides as reactants and converting these isolated, individual RNA monomers into a single RNA polymer which is complementary[2] to the DNA section being read. While the RNA polymer is being generated the original promoter-bound ligands may remain bound, or they may dissociate. If they remain bound, the RNA synthesis process may repeat, either before the first has finished, meaning two RNA molecules are being built concurrently, or some time after. The rate at which this re-starting occurs depends on a number of factors including nucleotide availability, the ligand species, cell state etc. This is the process of **transcription** - transcribing the information, previously stored as a DNA sequence, into the related medium of RNA. Once this has finished, we are left with a newly formed RNA molecule, the mRNA, and this mRNA molecule can then associate with a ribosome.

Ribosomes are massive, complex molecular superstructures which act as protein factories, taking in an mRNA molecule and amino acid monomers and producing an amino acid polymer (protein [3]. As in an assembly line factory, they have a conveyor belt like organisation, where the mRNA

[1] A ligand in this context is a small biological species (protein) which binds to a macromolecule (DNA)

[2] The concept of complementarity is non-relevant to this discussion, although is absolutely crucial to the process. For additional details see [66]

[3] The distinction between peptide, amino acid polymer and protein is subjective. A protein is an amino acid polymer, although in addition to the order of the amino acids (the primary structure) it includes additional information

molecule enters into the ribosome through a single entry point and is pulled in a stepwise motion through the ribosome. The ribosome scans the RNA as it passes through, and for every three RNA nucleotides translates this information into one of twenty amino acid monomers. That amino acid is obtained from the ribosome's local environment and is bound to an ever growing chain of amino acids. This is the process of **translation** - translating the information from the nucleotide medium into the protein medium. Once the full mRNA molecule has passed through, the ribosome releases this newly formed polymer, which goes on to form a protein through a number of post-translation steps, including folding and potentially chemical modification.

1.3.2 Description of Biochemical Systems

As a result of its central role in biology, traditional biochemistry has developed increasingly sophisticated techniques for analysing protein-protein interactions. As this research has progressed, more and more data regarding the network of proteins involved in various processes in the body can be summarized, and using graph theory based semantics and syntax these networks can be visualized and analysed [11].

Protein Protein Interaction (PPI) networks can be defined as undirected graphs, where nodes represent proteins and edges an interaction between those proteins[11]. Such graphs provide a visual, if highly simplistic way to represent the possible interactions in a cell, and databases such as KEGG are beginning to amass relevant information describing these interactions[57]. One of the primary drawbacks of PPI networks is the lack of directionality associated with the interaction, often because this metric is non-relevant. Biochemical cell signalling pathways can additionally be simplistically described as specialist directed PPI networks, specifying species involved in communication as nodes, and with the direction of communication as directed edges. With both of these descriptions we lack any kind of dynamism in our model. While adding weights to directed edges indicating relative speed of a pathway or strength of an interaction, the values associated with these interactions or reactions vary significantly with a wide range of factors. How rates of reactions vary has been the subject of extensive research over the last fifty years, and today reaction kinetics is at the heart of almost all biochemical processes. By understanding how fast a process is occurring, and what factors determine that speed, new insight into the mechanism behind the process can be explored. It is this kinetic modelling and analysis our work focusses on.

1.4 Kinetic Modelling

Much of the early organic chemistry relating to reaction kinetics originated in Germany in the late nineteenth and early twentieth century. There are a number of different ways reaction kinetics can be modelled, such as Michaelis Menten kinetics[29] or Hill kinetics[31]. This project focusses on using mass action ratio based kinetics, initially described by Guldberg and Waage in the late nineteenth century[28]. This is the simplest kinetic scheme (outlined below) and provides a straightforward starting point for simulations. Additionally, mass action kinetics have been used in previous studies on model systems similar to the ones developed, and it seemed prudent to maintain some kind of continuity, allowing the comparison between the results of our simulations with previously done ones. Although a critical underpinning, we do not focus on thermodynamics here, as it is considered beyond the necessary scope for this introduction.

The mass action ratio is the simplest measure for rates, and ties thermodynamic and kinetic analysis together. It provides a formal and mathematics description of the basic intuitive idea that the more of a reactant you have, the more products you generate, and that this can only happen at a maximum rate. Consider the equation below. Here A and B are reactants and C and D are

in terms of how it is folded (the secondary, tertiary and quaternary structural information). A peptide is a shorter, typically unfolded amino acid polymer, between two and around forty amino acids, although there is no definite cut off point)

products, which is to say we combine A and B and they form C and D. a,b,c,d are the *stoicheometric coefficients* of the species, and relay information regarding the ratio of molecular species between one another. Note these coefficients are specific to this equation, though not unique.

$$aA + bB \leftrightarrow cC + dD \tag{1.1}$$

With this setup in mind, the mass action ratio defines an equilibrium constant K_{eq} as shown below;

$$K_{eq} = \frac{[C]^c[D]^d}{[A]^a[B]^b} \tag{1.2}$$

Note that here, as in standard biochemical and chemical notation, [A] denotes the concentration of species A. The K_{eq} parameter, described here in terms of species concentration and stoichiometric coefficients can also be described in terms of the forward and backwards reaction kinetics. The reaction shown in equation 1.1 in fact describes two reactions

$$aA + bB \rightarrow cC + dD \tag{1.3}$$

$$cC + dD \rightarrow aA + bB \tag{1.4}$$

For each of these, the rate constant for the forwards reaction (as there is no backwards reaction in these irriversible reactions) is traditionally denoted k_1 or k_f (for equation 1.3) and k_{-1} or k_r (for equation 1.4). These two constants related to K_{eq} as follows;

$$K_{eq} = \frac{k_1}{k_{-1}} \tag{1.5}$$

Through subsitution, we can then derive the following equation;

$$k_1[A]^a_{eq}[B]^b_{eq} = k_{-1}[C]^c_{eq}[D]^d_{eq} \tag{1.6}$$

Which says that at equilibrium, the product of the concentration of reactants multiplied by the forward reaction constant is equal to the the product of the concentration of products multiplied by the backwards reaction constant.

We can further, then, expand these equations to the following equation

$$reaction\_rate = k_1[A]^a_{eq}[B]^b_{eq} - k_{-1}[C]^c_{eq}[D]^d_{eq} \tag{1.7}$$

At equilibrium *reaction_rate* is 0. However, at non-equilibrium this equation gives a simple reaction_rate equation which follows the mass action laws of chemical kinetics. It is upon this law we base our reaction laws in the models. The units of the k_1 and k_{-1} depend entirly on the euqation at hand, as we must ensure both $k_1[A]^a_{eq}[B]^b_{eq}$ and $k_{-1}[C]^c_{eq}[D]^d_{eq}$ are of the same units. A detailed dicussion is not required here, but to subtract one value from the other they must be the same type of information. As a result, kinetic constants are tailored to give us the correct values.

1.4.1 Time-course Modelling

With a basic description of mass action kinetics, the equations behind some commonly used kinetic models, various method for describing a system's progression through time exist. We can define a model scheme and a simulation scheme. The model scheme defines how the system is described at the starting point, and the mathematics behind how new values are calculated. The simulation scheme provides an algorithmic solution to solving variables defined by the model scheme, where such a scheme can be deterministic or non-deterministic.

The simplest model scheme is simply one based on mass action kinetics, where each reaction in the system is described as above, with the reaction rate based purely on the reactants and products. Such schemes fail to model certain systems correctly, but do provide a fast and effective starting point for building models.

Beyond a basic mass action kinetic scheme is the classic and simple example from Leonor Michaelis and Maud Menton's 1913 Michaelis-Menten kinetic model for enzyme kinetics[50]. Enzymes are biological catalysts, and facilitate a chemical conversion of one or more reactants into one or more products without themselves being destroyed. In the Michaelis-Menten model, which has it's foundations in mass action kinetics, an enzyme's reaction rate is determined by a number of factors, including reactant, product and enzyme concentration, the rate of catalysis, and the rates of binding and release of both reactant and product. Michaelis-menten modelling gave rise to the steady-state system description, whereby enzymes reach an equilibrium which responds to increasing levels of reactant or enzyme. While initial Michaelis-Menten kinetics focussed on simple systems, the scheme was subsequently extended to include the effect of reaction modifiers (both activators and inhibitors), leading to complex but effective equation sets which provide a good approximation for a number of systems. A Michaelis-Menten based steady state system provides a good model for many discrete biochemical pathways, especially those relating to metabolism. However, it is ill-suited to the unidirectional and non-linear flow of information associated with signalling.

Hill kinetics were developed as a method for describing cooperative binding, originally for a ligand binding to a macromolecule[31] but can more generally be applied to general reactions where reactants bind in a cooperative manner[29]. Cooperativity is the affect of multiple identical ligands binding to one species, where the binding of each successive ligand affects the affinity of subsequent binding, both positively (higher affinity) or negatively (lower affinity). In a combination of Hill and Michaelis Menten kinetics, Monod Wyman Changeux (MWC) sigmoid kinetics combine the cooperative of Hill kinetics with a Michaelis-Menten structure to take advantage of multi-subunit enzymes where catalytic rate subunit state affects the enzymes overall activity.

π-calculus (or process calculus) provides an additional tool, where a small number of logical rules, symbols and parameters define the allowed interconversion between species, the species themselves, their compartments, and all parameters associated with reactions. Developed for modelling complex computational systems, deterministic process calculus has had some biologically relevant success[44], although more research has been done into stochastic process calculus (or Stochastic Process Algebra (SPA)). BIO-PEPA[16] provides the relevant semantics and syntax to describe biological systems system, making up for some of the shortcomings of PEPA. PEPA uses an underlying Continous Time Markov Chain (CTMC) model to provide a stochastic component based on a negative exponential distribution. By using logical descriptors as opposed to hard (real) numbers, a system can be described in qualitative manner more akin to much of the data collected by experimental approaches. While BIO-PEPA has been used with some success for simulating biological systems [27], [58] it suffers from a lack of support in terms of user base, software, and models. Despite previous work in our group using BIO-PEPA, after a conclusive review of the state of the art[4] the decision was made to move away.

[4]Included in supporting information

Petri-Nets provide another method for discrete and parallel systems description. They are based on places and transitions, with arcs which connect the two. Places represent objects, and transitions how objects are inter converted. Each place may hold zero or more tokens, where tokens represent the number of objects in existence.

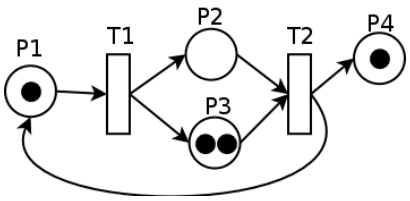

Figure 1.3: Simple diagram of a petri net P1, P2, P3 and P4 are places, with T1 and T2 transitions between them. The solid black circles are tokens [70]

A transition's activity depends on the arcs connecting places to that transition, and the number of objects at a connected place. Upon a transition, tokens are removed from the source object and added to the destination object, where the number of tokens added or subtracted depends on the weight of the arc. It requires integral values for tokens, and a number of both metabolic and signal-transduction based systems have been based on Petri nets [26]. However, despite some success they have had limited results with larger systems, providing an over simplification which inhibits the description of some biological events.

The aforementioned model schemes set out frameworks to describe the components of a system. We touch on a number of the most commonly used approaches, although there are many more not considered here. A system described by one of these setups can then be dynamically simulated using a number of different approaches.

1.4.2 Deterministic Modelling

Deterministic modelling is the original form of biochemical simulation, and provides a mathematical description which yields the same result each time the model's values are calculated over a time course. The most commonly used mechanism for such simulations are Ordinary Differential Equations (ODEs), where an ODE is solved to describe a species' concentration through time.

Petri nets can be solved by ODEs[23], although more typically are evaluated simply based on integral values and progressive basic calculations along a continuous time course for firing. This provides a deterministic implementation of Petri nets, although a stochastic implementation is also possible[9].

1.4.3 Non-deterministic (stochastic) Modelling

Following from the previous section, non-deterministic modelling provides some random element to a simulation, where the same result is not generated if the system is simulated multiple times. The underlying algorithm for many simulation techniques in chemical and biochemical stochastic models is frequently based on Gillespie's algorithm [24] [25].

When Gillespie's algorithm is not possible then some interface to a Markovian system is typically used. A Markov chain is a system with states which transition between one-another in such a way that the following transition depends solely on the current state - i.e. the transition pathway is memoryless. While a Markov chain has discrete time steps which trigger a (potential) change in state, a CTMC has this memoryless property, but additionally the trigger for state change is not a discrete value, but instead a negative exponentially distributed delay on the action [32]. This negative exponentially distributed delay means implicit probabilities of transition can be derived from the the distribution. CTMC are difficult to work with directly so a number of systems have

developed, essentially as high level wrappers to underlying CTMC model.

1.4.4 Project Approach

For this project, we have opted to begin with a simple approach and use MATLAB based ODE simulations. This decision was motivated by a number of factors. MATLAB provides a widely used and well defined interface for running biological simulations. The MATLAB SimBiology package[49] is well defined, and includes a number of relevant tools, such as automatic model checking, graph generation and an easy to use GUI. However, in addition these tools, it provides a powerful scripting language, allowing the running of complex simulations and analysis in an automated manner. In a previous dissertation, BIO-PEPA was used to carry out simulations, and while the simulations themselves generated interesting data, the lack of a scripting back-end made this project prohibitively difficult to develop into a high-throughput system[8]. By using MATLAB we avoid these issues, getting the best of both worlds, a convenient user interface, a powerful numerical engine, Linux/Mac/Windows portability, and a well developed scripting and programming language.

1.5 Monte Carlo Parameter Estimation

Parameter estimation is the process associated with taking a biological model and through some means generating system defining parameters, which when applied to that model allow it to behave as expected. This "expectation" is based on a comparison of the simulated data with experimental data, for example comparing the protein concentration time course in a simulation with experimentally collected results. The data generation can be done in a number of ways. Parameters can be initially estimated based on comparable systems, and then fine tuned to generate results in line with experimentally derived data. This typically yields one a single parameter set. An alternative method is to use Monte Carlo simulations to generate a random set of parameters, import these parameters into the model, run a simulation with that parameter set and then compare the results with experimental data. Where a parameter set has generated results which are favourably comparable with empirical data, that parameter set is saved, and the process is repeated.

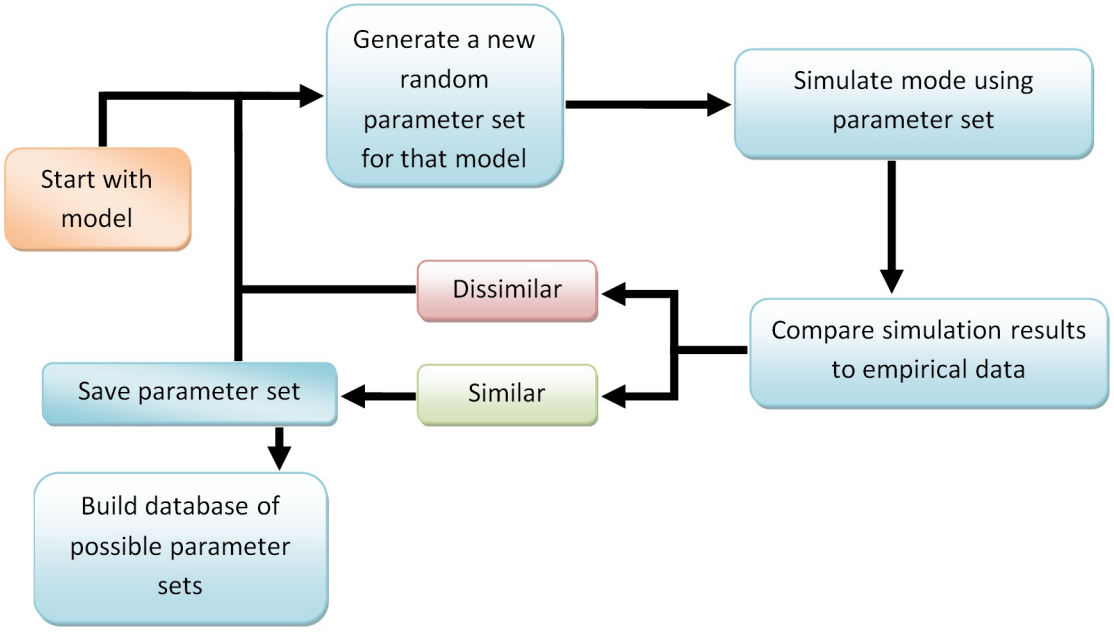

Figure 1.4: Schematic diagram describing the Monte Carlo parameter estimation algorithm

One factor which is often glossed over with regards to parameter estimation is the concept of parameter set scope. More specifically, (working under the assumption that the model is perfect) even if a parameter set is generated which produces results in line with empirical data, this does not mean that the values generate are the real rate constants. They simply represent a set which when used in a simulation generate the expected results, but there may well be many more sets. The number of possible sets which can generate identical results depends somewhat on the system. For example, in a pathway with one hundred species where empirical data exists for just three, there will be more parameter sets which can generate the same results for those three species than there would be if there were empirical data for all one hundred. However, in both cases it is likely (although admittedly not a certainty) that there will be many parameter sets which yield identical results. While this is true for Monte-Carlo simulations, it also holds up where parameters from an apparently similar model are introduced and fine tuned. Just because two systems appear similar (assume similarity is in terms of pathway topology and/or constituents) there is no guarantee the underlying mechanics bare any resemblance to one another. While superimposing parameters may be tempting (and often correct) this new parameter set is not necessarily "the" correct set of values - rather, it represents a member of a larger set of possible parameters.

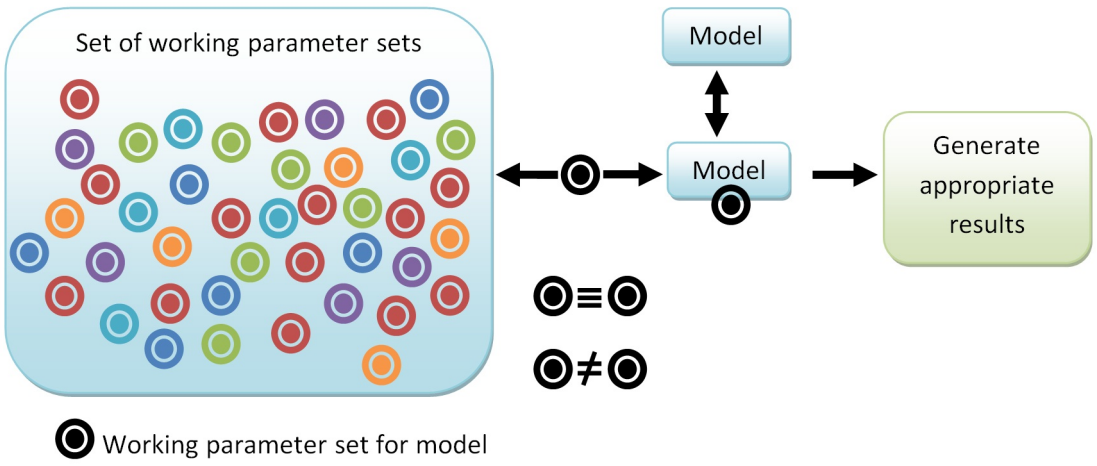

Figure 1.5: Graphical overview of the concept of multiple sets of functionally equivalent but numerically disparate parameter sets. Note, black concentric circles represent any of the possible parameter sets in the set of working parameter sets

In assessing parameters generated through Monte-Carlo simulation, therefore, there is a need to compare the working[5] parameter sets with one another. If, for example we are able to generate one hundred working parameter sets, and in all of these sets one of the values remains the same throughout, it is likely this specific variable is crucial, and that this consistent value represents potentially a real number. By analysing statistical indicators between equivalent parameters from working sets, a better picture of a model can be obtained. Indeed, if there appears to be no correlation, there may be a fundamental problem with the model. However, an in depth analysis of parameters sets generated through Monte-Carlo simulations goes beyond the scope of this paper.

In addition to the concept of sets of parameter sets, we must also accept that our model is a simplification of an incredibly complicated process. Ideally, such a model is biologically relevant if it captures the key determinants in a system. For example, if a signalling pathway is largely controlled by ten proteins, all of which are in our model, then the fact that another five hundred proteins have a minor role in regulating that pathway may not have a major impact of the final outcome. Bearing this in mind, however, it is important to remember that any numbers generated by the simulation are simply rough estimates, and not hard numbers.

[5]Working here is used to describe a parameter set which when applied to a model and that model simulated, the results generate data which is comparable with experimental results

1.6 Systems Biology Markup Language

SBML[34] has (arguably) become the standard format for describing biological systems. It combines a well defined specification with a complete syntax for describing models, as well as a number of highly developed APIs in a range of languages (C++, Java, C#, Python and MATLAB), each of which has extensive documentation. Although defined in a language agnostic manner, it's typical implementation is in an Extensible Markup Language (XML) based format. XML is a standardized mark up language for encoding information into a consistent and machine readable format[13] and is an ideal format for storing structured, periodic data such as that defined by the SBML specification.

Major editions of SBML are called levels, with minor updates termed versions. For this project we focus on level 2 version 1 (L2V1), however, higher levels and versions can be mapped down, although lower levels/versions cannot be mapped up. We have chosen to focus on L2V1 as it represents the majority of publicly available of SBML models. Many models are available in higher, more advanced versions, but in a number of repositories later formats can be automatically converted to earlier ones for downloading [56], meaning L2V1 acts as the lowest common denominator for virtually all models.

Through ever increasing use in systems biology projects combined with it's open source nature, SBML has developed a large user base as well as a a number of online model repositories, and a significant number of different software vendors have integrated SBML import and export into their products. By taking advantage of the backwards compatible nature of the format developments, SBML provides a broad, flexible language for defining the features relevant to a specific model or system in a consistent manner, without a forced need for overspecification.

1.6.1 Specification Summary

A complete overview of the specification can be found here[34]. An SBML model is structured as lists of one or more of the ten key components outlined below. We included a brief description of each of these components and their role. It's important to note this is a highly superficial overview - the specification document is succinct, clear, and well written and manages to encompass 167 pages. Each component is a list of elements - these lists simply act as containers for the elements stored in them and provide a logical interface for iterators. For each element, assume a non unique name (for easy reference) and a unique ID (for internal model referencing) unless otherwise suggested.

- **List of function definitions** *(optional)* - To add clarity to the specification, users can define commonly used functions here and then refer to them throughout the model
- **List of unit definitions** *(optional)* - Where units are not basic unary descriptors (moles, litres, seconds etc.) the user can define multiple-component units (mole/sec, litre^3) here and then refer to them throughout the model
- **List of compartments** *(optional)* - Individual compartments can be defined here. Typically this may only include the cytoplasm, but may also include the nucleus, vesicles, or any other compartments. Compartments include a size, the compartment's units, and a boolean value to show if their size is held constant through a simulation
- **List of species** *(optional)* - Each species, such as proteins, ions, enzymes, small molecules or conceptual items (such as "protein synthesis") is represented as a species. Species have a starting amount or concentration, units, associated compartment, and a number of other attributes
- **List of parameters** *(optional)* - Parameters provide a simple element which stores some value. This value can change, or can remain constant, and can have units associated with it

- **List of initial assignments** *(optional)* - Initial assignments are a little different from the elements already encountered - they lack a name or an ID, instead providing a mechanism to set the initial value of a species where that value depends on other factors which cannot be predetermined. They can be seen as;

$$species\_ID = initial\_assignment(p_1, p_2, ...p_n) \tag{1.8}$$

  Where p_x can be constants or variables determined by an initial assignment, a parameter from the model, other species' initial concentration, or a function with input values

- **List of rules** *(optional)* - Rules provide a way to dynamically calculate a parameter rather than have it pre-determined. There are three kinds of rules;

  - *Assignment rule* - Used to express an equation which sets a variables value (such as a parameter, species concentration, or compartment size)
  - *Rate rule* - Used to express the rate of change of a variable (such as a parameter, species concentration or compartment size)
  - *Algebraic rule* - Used to calculate a numerical intermediate or temporary value

  Like initial assignments, rules do not have a name or ID, but instead simply act on an element, essentially providing an optional extension for that element

- **List of constraints** *(optional)* - Constraints are a mathematical description of model assumptions which the model must remain under. If during a simulation the constraint is no longer satisfied, then that simulation is deemed invalid. A constraint might be that no species concentration can become negative (for example)

- **List of reaction** *(optional)* - Reactions are by far the most complex element in the SBML specification. They consist of a number of sub-elements defined below. Overall they provide an element which describes a reaction between one or more species ("reactants"), and their conversion to one or more species ("products"). The basic reaction object, as well as a name and an ID, has a boolean value to determine if it is reversible or not, and the compartment it occurs in

  - *List of reactants* - A list of species which are the reaction's reactants
  - *List of products* - A list of species which are the reaction's products
  - *List of modifiers* - A list of species which are the reaction's modifiers (both activators and inhibitors)
  - *Kinetic law* - The kinetic law includes an equation to determine the rate of the reaction. This equation can include predefined functions, parameters and species. However, any species used to define the rate must be included in one the aforementioned lists (reactants, products or modifiers).

  In addition to global parameters (defined in the list of parameters) reactions can have their own local parameters, in a list of parameters object associated with the kinetic law. These parameters take precedence over global parameters.

- **List of events** *(optional)* - Events describe discontinuous, discrete changes made in response to a certain state in the model. They include a trigger element, which defines what causes the event to be fired, a list of event assignments, which define what assignments occur in response to the trigger, as well as a number of other elements such as priority and delay objects which allow fine tuning.

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2" xmlns:html="http://www.w3.org/1999/xhtml" level="2" vers

  <model id="mw9429f8ce_d823_4398_bbab_177d4f795a17" name="cytosol">

    <listOfUnitDefinitions>
      <unitDefinition id="unit7" name="unit7">
        <listOfUnits>
          <unit kind="metre" exponent="3" multiplier="1" offset="0"/>
          <unit kind="mole" exponent="-1" multiplier="1" offset="0"/>
          <unit kind="second" exponent="-1" multiplier="1" offset="0"/>
          <unit kind="dimensionless" multiplier="1.66666666666667e-005" offset="0"/>
        </listOfUnits>
      </unitDefinition>

    </listOfUnitDefinitions>
    <listOfCompartments>
      <compartment id="comp_main" name="main" size="1" units="liter"/>
    </listOfCompartments>

    <listOfSpecies>
      <species id="main_cytosol_MK2" name="MK2" compartment="comp_cytosol" initialConcentration="0"
       substanceUnits="mole" spatialSizeUnits="liter"/>
    </listOfSpecies>

    <listOfParameters>
      <parameter id="main_cytosol_Rxn1534_Kf" name="main.cytosol.Rxn1534.Kf" value="99999300"
      units="unit7" constant="false"/>
    </listOfParameters>

    <listOfRules>
      <assignmentRule metaid="repeatedAssignment_mw83aa826b_f165_4028_a6f9_121ebb9c248b"
      variable="main_cytosol_Rxn1534_Kf">
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <ci> main_P_kf_LPS_TLR4 </ci>
        </math>
      </assignmentRule>
    </listOfRules>

    <listOfReactions>
      <reaction id="main_cytosol_Rxn1271" name="Rxn1271" fast="false">
        <listOfReactants>
          <speciesReference species="main_cytosol_complex_MKK6"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species="main_cytosol_complex"/>
          <speciesReference species="main_cytosol_MKK6P"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns="http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci> main_cytosol_Rxn1271_Kf </ci>
              <ci> main_cytosol_complex_MKK6 </ci>
              <ci> comp_cytosol </ci>
            </apply>
          </math>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

Figure 1.6: Example of an XML SBML file structure, with the elements described above labelled for clarity. The content here is irrelevant, although included as a demonstration

This overview describes the ten basic components in an SBML file. It is clear that there is in fact significant overlap in terms of what can be done to achieve the same result. For example, an event can be used to set the initial concentrations (by defining a trigger at t=0), or alternatively an initial assignments could be used. This functional redundancy is not accidental. In the context of the rapidly evolving understanding of biological systems and the development of different ways to represent them, it is impossible for one single system modelling language to include all the required detail for all systems without introducing a significant amount of redundancy into the majority of described systems. SBML, therefore, provides a flexible core for describing the fundamental components of a system, which can then be built on by other software to include necessary detail where appropriate, or simply use the optional objects and attributes provided by the SBML specification. This framework does not require over determination where inappropriate, nor does it typically introduce significant redundancy into the model. By providing a relatively flexible specification, SBML provides a framework for structured system description without being overly restrictive in terms of what can and cannot be done.

1.7 Asthma

In this section we provide an overview of the molecular characteristics or asthma, with a focus on GC resistant asthma. Much of this content is relevant for our generation of the GC signalling pathway, although is not crucial to understand the project's development. We include it here primarily as reference material for those more biochemically minded.

1.7.1 Overview

Asthma is characterized as a chronic disease which affects the respiratory system, inflaming and narrowing the airways to induce breathing difficulties [54]. Typically, a trigger causes a spontaneous attack, which may dissipate rapidly (within hours or even minutes), or may last significantly longer (a period of weeks or even months). Currently affecting 300 million people, that number is predicted to rise by an additional 100 million by 2025[3]. While the vast majority of suffers keep the disease in check using inhalable corticosteroids, occasionally in combination with Long Acting β Agonists (LABAs), around 5% of those affected are resistant to this treatment. This small fraction generates a disproportionate amount of the costs associated with sever asthma[36]. By better understanding the molecular mechanism associated with this resistance, improved treatment regimes and more effective drugs could be designed to reduce its impact of the quality of life of individuals, as well as the cost burden burden associated with the resistant asthma.

1.7.2 Molecular Mechanism

The asthmatic triggers depends on the individual and the environment, but are typically associated with an allergic response, which may be caused or exacerbated by infection (bacterial or viral), or pollution. The systemic inflammatory response has been the subject of extensive research, and is not considered in detail here. In summary, upon detection of an allergen, mast cells in the tissue around the airways release granules packed with inflammatory mediators, including histamine, which interact with the cells in airway tissue (especially those of smooth muscle) to trigger constriction and an immediate reduction in airflow to the lungs. These mast cells also produce a whole host of other inflammatory signalling molecules (such as prostaglandins, leukotriens, kinins, Interleukin (IL)-4, IL-13 and chemokines) which promote muscular constriction and attract other immune cells such as neutrophils, lymphocytes and eosinophils. Additionally, lymphocytes release IL-5, macrophages produces Tumour Necrosis Factor (TNF)-α, and a variety of other molecules such as neutrophil chemokines, cytokines and growth factors such as IL-6, IL-11 are produced. These inflammatory signalling molecules all cause the surrounding cells to shift into an inflamma-

tory state, which is primarily regulated by the antagonistic actions of corticosteroids.

Inside the airway tissue cells, this recruitment and activation of immune cells and the expression of inflammatory signalling molecules triggers a number of processes which lead to the intracellular inflammatory cascade, and is not something considered here. Additionally, there is a demonstrable increase in the transcription and activation of a number of inflammatory transcription factors, such as the NF-κB and Activator Protein-1 (AP-1) families of proteins[4]. Both of these are activated by a whole range of inflammatory signalling molecules associated with the asthmatic response. NF-κB family proteins are responsible for the up-regulation in expression of a wide range of molecules, such as chemokines, cytokines, growth factors and enzymes[5], while the genes up-regulated by AP-1 are more typically associated with proliferation and survival.

This frequently leads to the chain reaction observed in asthma attacks (a sever and prolonged episode of restricted breathing caused by the inflammation) - the initial trigger leads to the transcription of the critical signalling molecules (such as cytokines, chemokines and interleukins), which in turn leads to the expression of those proteins involved in the intracellular inflammatory cascade. At the same time, the signalling molecules cause an increase in the expression and activation of transcription factors such as NF-κB and AP-1.

NF-κB is activated by a wide range of different stimulatory elements through a number of well defined pathway[6]. Active AP-1 is a heterodimer of *Jun* and *Fos* proteins, and characteristically binds to the TRE (TBA response element). The formation of the active heterodimer is facilitated by the Mitogen Activated Protein Kinase (MAPK) family kinase c-Jun N-terminal Kinase (JNK) through a number of phosphorylation events. NF-κB and AP-1 go on to up-regulate transcription of inflammatory proteins as well as those signalling molecules, causing a positive feedback loop which maintains a state of inflammation in the tissue. In addition to this transcription/expression cycle, chromatin remodelling, typically consisting of histone acetylation to relax DNA tension and facilitate transcription, triggered by CBP (CREB (cAMP (cyclic Adenosine Monophosphate) Response Element Binding Protein) Binding Protein) also contributes to inflammatory protein expression[64].

Inflammation is mediated and reduced by corticosteroids. These signalling molecules signal between cells (in the extracellular environment), and originate from the adrenal cortex. Synthetic versions provide a much higher dosage in a localized manner through an inhaler, rather than relying on the circulatory system. They bind to Glucocortocoid Receptors (GR), found in almost all cells, which then dissociate from a Heat Shock Protein (hsp)-90 chaperone protein to form homodimers. The homodimeric protein can now diffuse into the nucleus, where it binds to a Glucocortocoid Response Element (GRE) in the DNA, where the AF-1 and AF-2 (Activation Function) domains alter DNA transcription. This is where the interplay between the pro and anti inflammatory signals occurs - the ligand bound GR dimer binds and represses transcription of inflammatory and immune genes, effectively counteracting the activities of NF-κB and AP-1.

Corticosteroids bind to NF-κB directly and indirectly, causing a reduction in the transcription factor's ability to up-regulate transcription of inflammatory proteins. Additionally, corticosteroids repress MAPKs such as Extracellular Regulated Kinase (ERK) and JNK. By inhibiting JNK activation, corticosteroids block the formation of the active AP-1 heterodimer, meaning corticosteroids implement a rapid response mechanism to alleviate inflammation both by directly reducing the expression of inflammatory molecules and by reducing the transcription and activation of pro-inflammatory transcription factors. Dexamethasone, a potent synthetic corticosteroid, was also shown to induce MAPK phosphatase-1 (MKP-1) (an inhibitor of MAPK) and as a result reduces p38 activity, a MAPK frequently associated with inflammation.

GRs are phosphoproteins, meaning their state is altered by phosphorylation. With five sites for phosphorylation (Ser113, Ser141, Ser203, Ser211 and Ser226) phosphorylation provides an effective

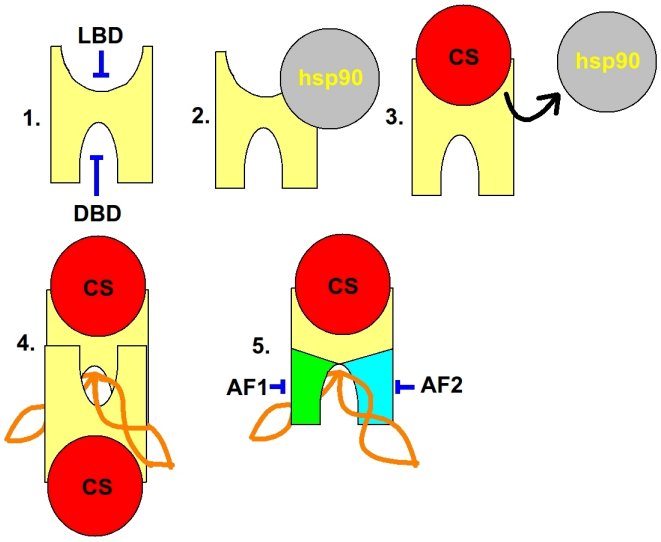

Figure 1.7: 1. Schematic diagram of GR with LBD (Ligand Binding Domain) and DBD (DNA Binding Domain) highlighted. 2. hsp90 binds to the GR and stops it dissociating from the cell membrane 3. Corticosteroid bind as ligands and trigger hsp90 dissociation, releasing GR from the membrane 4. Two free GR bind together as a dimer, and can the bind DNA 5. The AF-1 and AF-2 domains in a monomer (single GR) are highlighted here.

mechanism to control GR activity. Agonist or antagonist binding to GR effects how the receptor is phosphorylated [68], and the phosphorylation state has been associated with a range of functions. Phosphorylation has been shown to affect DNA binding in a promoter dependent manner [18][69], and Ser211 phosphorylation affects ligand binding, nuclear localization and and GR activation[18]. Conversely, in another study Ser211 phosphorylation seemed to increase GR functionality [41]. MAPK, Cylcin Dependent Kinase (CDK), Glycogen Synthase Kinase 3 (GSK-3) and JNK are all implicated in the phosphorylation of GR, although the the role (if any) of p38 remains unclear. In HeLa cells (a standard immortal human cell-line used in a wide array of scientific experiments), up regulated p38 lead to a reduced GR activity through some effect on the ligand binding domain. However, mutation in the p38 gene had no impact on this functionality, suggesting a possible downstream activity relating to GR chaperone proteins or co-activators[62]. However, in human and mouse lymphoid cells p38 has been shown to directly increase phosphorylation of GR[51]. Essentially, this appears to be a highly complicated tissue and species specific mechanism, where a general consensus on the role of enzymes and phosphorylation on activity is yet to be determined.

1.7.3 Corticosteroid Resistance

Corticosteroids provide a fast and effective treatment for asthma, and represents the foundations of the standard regime for managing the disease at a severe level. However, in a small number of patients even a high concentration of oral corticosteroids is ineffective [48]. Alternative therapies, such as targeting IL-4, IL-5 and IgE (immunoglobulin E) has been investigated, but clinical trial results were not encouraging [12] [42] [55]. Better understanding corticosteroid resistance in asthma is crucial in developing new and effective treatment strategies for those affected. From a clinical perspective, CSR asthma is typically diagnosed when the condition fails to improve despite long-term (14-days) use of oral corticosteroids. On a molecular level, a reduced suppression of IL-4 and IL-5 mRNA in bronchoalveolar tissue has been reported, as well as a greater number of cells expressing IL-2, IL-4 and IL-13, indicating that cytokine expression profiles may contribute towards the resistance [47]. Peripheral Blood Mononuclear Cells (PBMCs) - blood based cells with a round nucleus such lymphocyte, monocyte, macrophage, but not neutrophil or eosinophils as these contain multilobed nuclei - display reduced sensitivity to corticosteroid suppression, meaning in CSR asthma patients corticosteroids have a reduced impact of PMBC cytokine release.

Cortecosteroid resistance appears to be mediated in patients by a number of distinct mechanisms, reflecting the complexity of this regulation. In acquired CSR patients, mechanisms include reduced GR expression, reduced affinity of corticosteroids for GR or a reduced affinity of GR for DNA. Indirect mechanisms include a decreased expression/activity of co-repressor proteins and an increase in NF-κB or AP-1 expression. Patients are not GC deficient[20], with serum GC lying in normal ranges[45]. Similarly, no reduction in the gastrointestinal absorption of GC[46] is seen. Patients display elevated levels of pro-inflammatory interleukins[47] in the bronchial tissue, suggesting an inability to effectively respond to the actions of GC. These elevated levels may be caused by p38 MAPK phosphorylation of GR[35], activating the inflammatory response pathway. Additionally, a distinct reduction in p38 MAPK phosphatase expression is seen in CSR patients after exposure to GC [23], and dysregulation of JNK and AP-1 both appear to have some role in the desensitization to GC[36]. We end our overview of the biological mechanism of GC resistance here, although the extensive reference material highlighted in the section provides ample discussion on the topic. It is clear that this is a highly complex issue, which does not have a "black and white" answer. Indeed, it seems likely that a number of discrete molecular conditions would give rise to the same physiological lack of response to GC, although many of these diverse mechanisms may in fact be mediated by a small number of interfaces between crucial pathways.

1.7.4 Pathway Crosstalk

Understanding signalling crosstalk between different pathways in the context of GCs signalling may provide crucial information to help understand the molecular mechanism of CSR asthma. There is experimental evidence to suggest that in some forms of CSR patients, the cause of this resistance is a genetic defect which causes a disruption in normal crosstalk between pathways[4]. Historically, rare genetic anomalies have often provided insight into complex molecular mechanisms, so it seems plausible that while dysregulated crosstalk can have serious effects, under normal conditions crosstalk provides an underlying homeostatic mechanism which normally maintains a careful balance between pro and anti-inflammatory signals. If this is the case, then treating CSR patients with GC will have no impact on their well being. However, combining GC with components to offset or correct cross-talk associated problems may allow these traditional regimes to be effective. In the preceding section, we mentioned the possible impact of JNK, AP-1 and p38 MAPK on GC signalling, and it is this set of key mediators we focus on here.

Crosstalk appears to occur both at the signalling level and at the gene regulation level, meaning both aspects should be considered[53]. However, the intricacies associated with the crosstalk is poorly understood. There is a tendency to study pathways as linear routes from trigger to effector, despite this being a poor representation of signalling systems. As a result, the nuances relevant to pathway crosstalk offered by many proteins is lost.

1.8 Preceding Work

This project is the evolution of two sets of work: a previous MSc project carried out by C. Baroukh[8] and a paper published by B. Hendriks, F. Hua and J. Chabot entitled "Analysis of Mechanistic Pathway Models in Drug Discovery: p38 Pathway"[30].

Hendriks et al. developed a model of the p38 MAPK signalling pathway shown below, and additional determined a range between which the parameters in the model would lie based on general values for biological reaction kinetics.

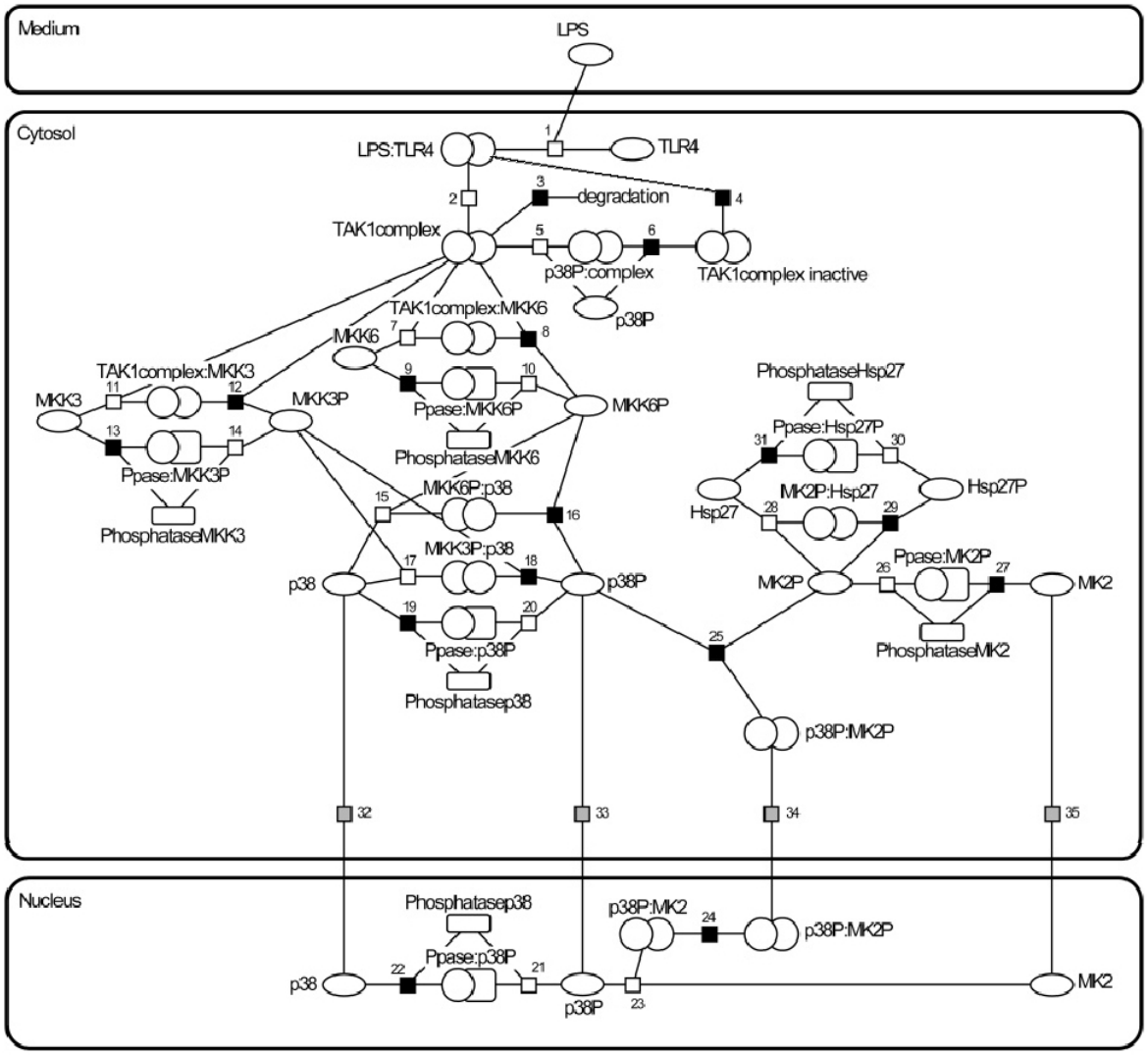

Figure 1.8: p38 MAPK model[30] (includes a minor updated - see section 3.2 for further details.

The Hendriks model was developed in the Teranode Design Suite (Teranode Corporation), exported to SBML and translated into JACOBIAN (Numerica Technology). Simulations for parameter estimation were then performed in JACOBIAN, and a number of possible parameter sets were generated. Our starting point for this project, was this p38 MAPK model developed by Hendriks et al, combined with the parameter ranges they had already determined.

Baroukh's dissertation was based on using some of the parameter sets defined by Hendriks et al with the BIO-PEPA language to simulate the p38 MAPK pathway using a BIO-PEPA generated model, running a version of Gillespie's algorithm. While this was successful, it became obvious that the overall lack of software support for the BIO-PEPA framework at this time meant that high-throughput simulation and analysis with BIO-PEPA was not-practical. Additionally, despite

having idealized parameters based on the model developed my Hendriks et al. the results of Hsp27P species concentration for a simulation with these idealized parameters using Gillespie's algorithm based on the BIO-PEPA model does not match the experimental data provided. In contrast, the JACOBIAN based ODE simulation gives a much closer match, indicating potentially a problem with the BIO-PEPA simulation framework. However, this has not been explored further, and my in fact reflect a flaw in the JACOBIAN model.

Chapter 2

SBMLIntegrator

Overview

In this chapter we overview one of the key problems encountered early on, and describe the approach and implementation of a software solution. The specifications regarding this problem are considered, general design concepts shown and the software engineering methodologies used are named and justified. We finish by describing the finished project, evaluating it in the context of our original goals, and discussing the future work associated with it.

2.1 Introduction

The ultimate aim of this project is to built a framework to evaluate the existence of crosstalk between two different pathways, the p38 MAPK pathway and the GC signalling pathway. Upon initial research into the tools available for SBML, it became clear there were no tools for integrating two SBML models together. There are a number of SBML model editors available, so although constructing three models (the isolated p38-MAPK pathway, the GC-signalling pathway and the combined pathways) would be possible, this seemed an unnecessary duplication of work. Moreover, if our pathways were significantly larger, building a new model by extending out an original model manually is a very time consuming activity, and one fraught with the risk of error.

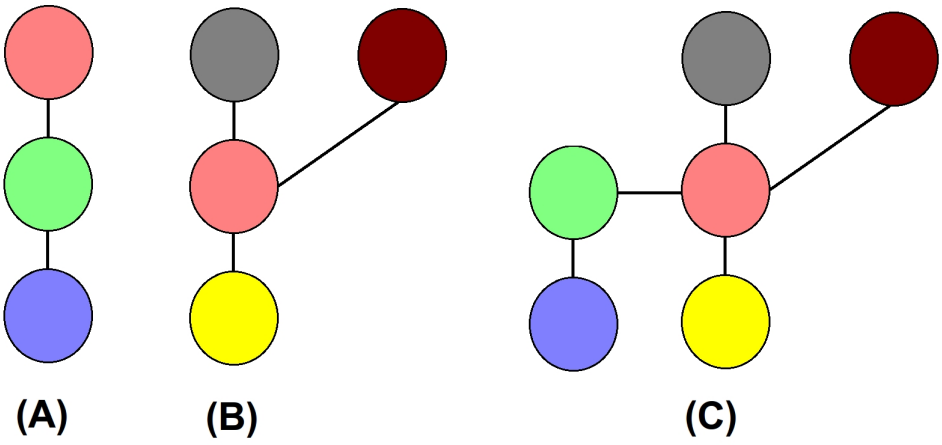

Figure 2.1: Example of integrating pathways **A** and **B** to form pathway **C**. Note that in the new pathway, we have replaced some identical elements with others to construct one, single system.

If we consider the SBML overview discussed in chapter 1, it becomes clear these models heavily utilize internal dependencies. If two pathways are being combined there will inevitably be a significant amount of overlap between those models, both in terms of species and compartments,

but also units. Additionally, there is a need to ensure element IDs are unique across both models. Some software check this, but many do not, leading to scoping conflicts. The ID variable is not enforced as unique in the SBML specification to allow for deliberate namespace ambiguity (e.g. for local parameters), however, in situations where two models are being combined it can lead to silent errors which disrupt a model while failing to generate any obvious symptoms. A tool to automate the integration of two models in a semantically and syntactically accurate manner would speed this process up immensely, and significantly reduce the risk of human error in what would be an incredibly long winded and tedious process.

A second motivation for developing an automatic integration tool is that biological understanding of systems often changes, leading to a need to update models or parameters. Such updates to a poorly constructed integrated system risk breaking the model - small changes can have significant impact across a model in a cascading effect. Typically the reason for integrating two models will be to investigate how they interact, which suggests a lack of detailed understanding of this newly formed integrated model, but a better overview of the two individual models. By ensuring integration is quick and easy, researchers can make changes to individual models and then re-integrate them, rather than navigating through the complex and convoluted task of editing an already integrated model in the context of one of its two submodels.

Ultimately, this relates to the concept of least resistance - if a tool is available to make integrating two models together easy, then this becomes an appealing research topic. Hundreds of models already exist in the BioModels Database[56], many of which could be integrated and investigated for crosstalk. The description of biochemical pathways as semi-linear isolated communication networks is largely an inaccurate representation, brought about because it provides an ideal way to teach and understand these pathways. Instead, pathways should be looked at as a smaller subcomponent or a larger multi directional and multidimensional network, and integrating two pathways together is a means of enlarging two single fragments of an overall signalling network into one, larger section of that same network. This is the role of SBMLIntegrator.

2.2 Specification

Considering the challenges discussed, there were a number of initial high level goals for this software;

Functionality
The tool should allow the integration of two models basic[1] in a flexible manner, where elements from one model can be added to, replace or be integrated with elements in a second model.

Ease of use
A tool should remove all possible semantic difficulties, and where it cannot do so should offer users the opportunity to make a decision with a number of options presented. The process of integration may not be trivial, however, the overall goal is to ensure any difficulty arises from biological questions, rather than those relating to SBML semantic or syntax.

Ease of installation
One thing which became clear during the initial overview of the various SBML software available was that in almost all cases installation was time consuming and complicated. Frequently, extra (often deprecated) libraries were required, which had to be manually configured, and a range of other problems. To avoid this our aim is that the user needs to simply run a single “Install” program once, which will install the libSBML library automatically, and then install the SBMLIntegrator.

[1]Basic in this sense means a model containing one or more of the following elements: Unit Definitions, Compartments, Species, Parameters, Rules and Reactions only.

Use of a configuration file
The process of determining how to integrate two models is (from a biological standpoint) difficult. By providing a simple configuration file where users can enter the specific parameters regarding their model integration, this abstracts much of the decision making away from a live program to one where the user has time to carefully consider their options. Additionally, a separate file avoids time consuming and repetitive interaction, especially if two models are integrated multiple times with small changes to integration settings but not to the species being selected. Ideally, the user should be able to write a configuration file once, and re-integrate two models instantly thereafter, assuming they do not change the way in which the integration occurs. If they do, however, this is simply a case of updating the configuration file.

Completed in a timely manner
It is important to bear in mind that this thesis is not primarily a software engineering project. Considering this, despite taking a thorough approach to the development of SBMLIntegrator, there is also a need to objectively prioritise the features required for this specific project. While generating a software which can integrate any generic models is key for the long-term success of SBMLIntegrator as an independent software tool, in the short term we have focussed here on basic SBML models, which exclude those which contain constraints, function definitions, initial assignments or events. However, the overall software structure will be developed so as to ultimately facilitate the introduction of this functionality. With necessity as the mother of invention, this software was born of a need to integrate two SBML models together, and the realisation that this may be a more general problem. However, developing a fully fledged software tool to deal with all eventualities was not a project objective as such, so while this tool does work for a number of generic cases, it is not all encompassing, although these limitations should be viewed in the context of this project, rather than in the context of the SBML specification as a whole.

2.3 Design

Based on the high level goals outlined above, we developed an overall initial software structure for the project. A schematic of this structure is outlined below.

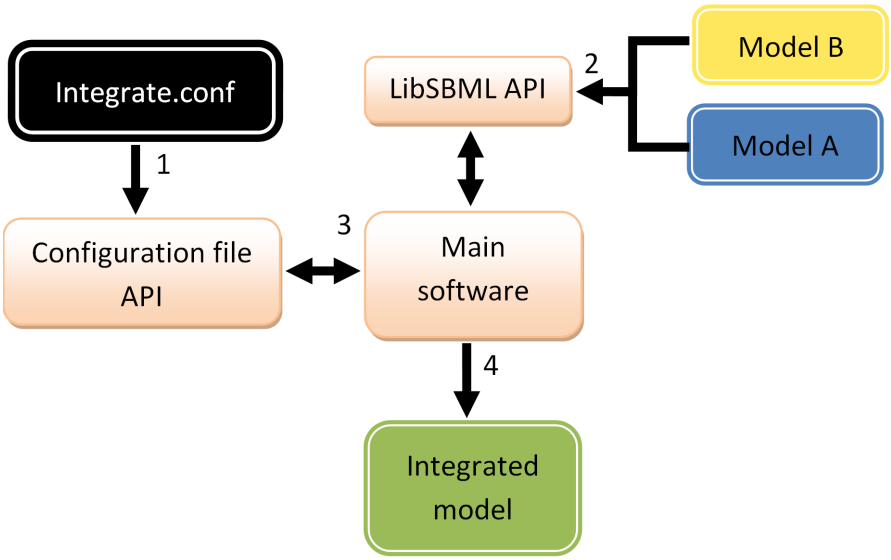

Figure 2.2: Simplified schematic of the software's architecture, with reference to how it interacts with external files. **(1)** The integration configuration file (discussed below) defines the integration behaviour, and this information is parsed by the configuration file API. **(2)** Models are read (and in fact written) by functions provided by the LibSBML API. This API is provided for us, and includes a range of functionality, as well as a complete set of containers with getter and setter functions for all SBML elements. **(3)** The configuration file API provides a transparent set of functions to the rest of the software for getting relevant data from the configuration file. **(4)** The main software then uses the parameters defined by the configuration file and the data imported from the two models to integrate them together and construct a new, integrated model

Figure 2.2 describes how SBMLIntegrator interacts with the three input files it requires. The two models are imported into the software, however, they remain in the containers implemented by the LibSBML API. Similarly, the configuration file is parsed by the configuration file API, which stores all the information from that file. The main software acts as a co-ordinator of the data in a controlled manner, but avoids the issues associated with data ownership, duplication and integrity. The LibSBML API containers are well implemented and provide perfect data structures. Similarly, we use a combination of LibSBML data structures and our own template structures to store the configuration file information. By abstracting the stored information into specific modules we help to clarify the flow of information. The process of integration is by it's nature complicated, with serious issues surrounding data duplication, data validity and data corruption. By ensuring we maintain a copy of all original data in a state where it cannot be changed while developing a new model in a stepwise, dynamic and deterministic manner, we provide fixed reference points for the integration process while implementing that integration in a logical and efficient way.

One of our earliest design decisions was to allow integration to proceed through the use of three models. These are best described by an example where we are integrating model A and model B together. In this example, model A is three times the size of model B. Therefore, we designated model A the **base model**. The base model is used as a template for the new model being created through the act of integration. The base model is copied, and this copy forms the **integration model**. Initially the integration model and the base model are identical. However, as we proceed through the various integration operations, we change the integration model by bringing in components from the **import model** (in this example model B). The import model is typically the smaller of the two, and from it we introduce new elements into the integration model. Neither the base nor import models are changed in any way, but instead act as static reference points for the process of integration use. We refer to the various models as base, import or integration models throughout this chapter. The three models are outlined below, where a basic overview of how integration proceeds.

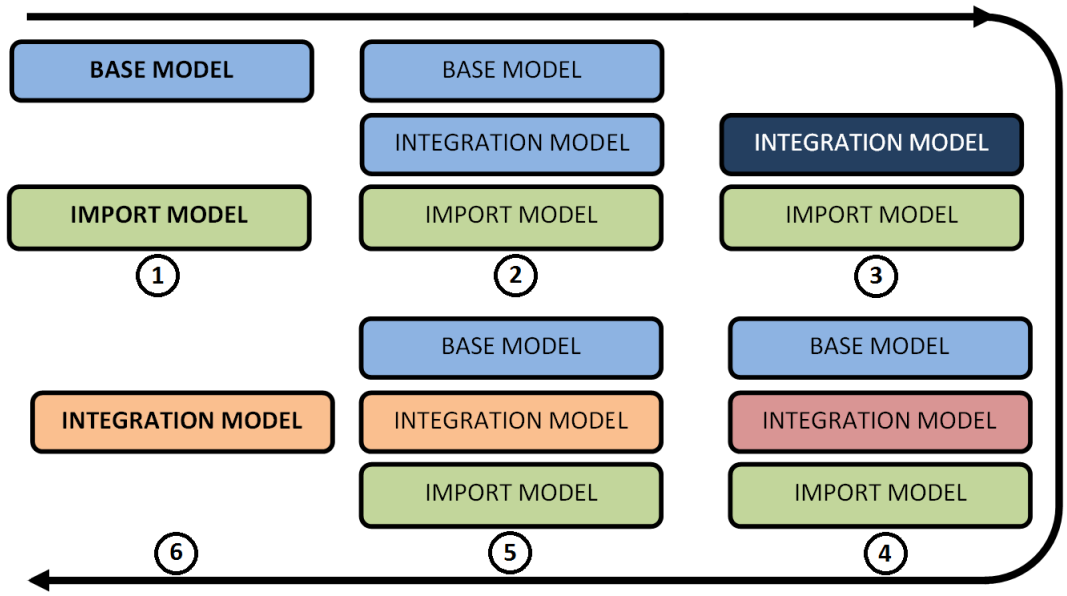

Figure 2.3: Overview of the progressive nature of integration based on the three models outlined previously. **(1)** We begin with the base and import models. **(2)** The base model is copied to form an identical integration model. **(3)** The integration process begins by taking new and unique information from the import model. **(4)** Following this, duplication and redundancy introduced by the action of importing new elements from the import model is removed. **(5)** Finally, an interactive integration of elements which contain overlap in their attributes for the new model is carried out. This may include elements such as new reactions which combine reactants and products of two different reactions. **(6)** The integration model is now finalized.

After identifying a good method for model representation, the next question regarding the SBMLIntegrator software was, "*What is the best way to integrate two models together?*". On the surface, this seems a fairly simple question, but as various approaches were investigated it became clear this was less straight forward than anticipated. There is a need to balance efficiency with usability and flexibility. By committing the user to a specific mechanism of integration we reduce their ability to do certain things, but equally, by allowing integration to be open and flexible we risk huge redundancy in the software, or worse, data inconsistency. After a number of theoretical prototypes built as flow charts and then evaluated with both user interaction and time/space complexity in mind, we developed the Import Replace Integrate (IRI) mechanism for integration.

2.3.1 Model Integration

Early in our research into methods for integration, it became clear there are three distinct operations which can be implemented to integrate elements from two separate models;

- **Import** - We can simply import (copy) an element from model A into model B.
- **Replace** - We can define an element in model A to replace an element in model B. This involves not only overwriting the element in model B, but additionally scanning through model B and changing any reference to the overwritten element to the new element from model A. For example if one one model contained Alice and the other Bob, and we replace Alice with Bob, then not only would both models now contain Bob (and neither Alice), but any references to Alice (such as "*Alice in wonderland*") would now be replaced by Bob ("*Bob in wonderland*")
- **Integrate** - We can integrate two elements together, combining aspects of the elements from both model to create a hybrid element. For example, if Alice wore red shoes and Bob wore brown shoes, we could integrate the two together, meaning Alice may now wear brown shoes (and Bob would still wear brown shoes as the integration only affects on of the party).

With these three operations in mind, we can discuss the semantics associated with IRI.

IRI integration is based on the fact that you can quickly predefine elements you wish to import from one model into another. Similarly, you can also quickly define elements from one model which you wish to use to replace elements in another. This process is the same for any element type - for example, conceptually, replacing Alice in model A with Bob from model B is the same operation as replacing MAPK in model A with ERK in model B. The *type* of component which Alice, Bob, MAPK and ERK represent is irrelevant. Moreover, there is only one possible outcome from this replacement operation. The same is true for the import operations - the outcome of importing the first three species from model B into model A will always be the same.

However, integrating two elements together is not uniformly comparable over all element types, and does not always yield the same result. Compartments have different attributes compared to species, and again compared to units, and so on. Additionally, where elements have a number of attributes we do not pre-determine how we integrate these individual attributes. For example with our previous Alice and Bob example, instead of selecting Alice's name ("Alice") and Bob's shoe colour ("Brown") we could have selected Bob's name and Alice shoe colour, or chosen to change everything or nothing. As shown below this gives four equally possible an correct outcomes of the same integration, where Alice and Bob are unique IDs, and each individual has two attributes, a name and a shoe colour;

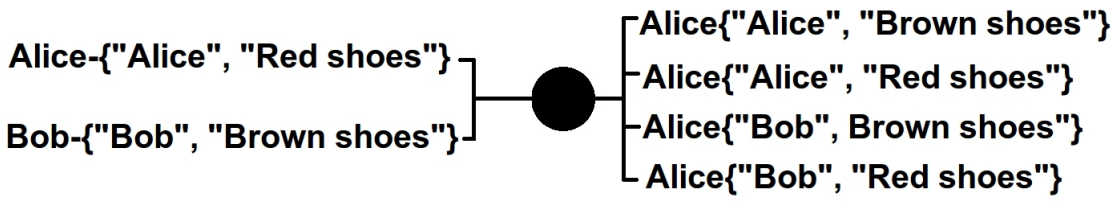

Figure 2.4: The four possible outcomes if we integrate Bob into Alice (note the directionality here)

Beyond this, it should be clear that the order we carry out our replacement, import and integration operations has profound semantic consequences. If we had previously replace all the references to "Red shoes" with "Brown shoes" in Alice's model our option for integration would be reduced;

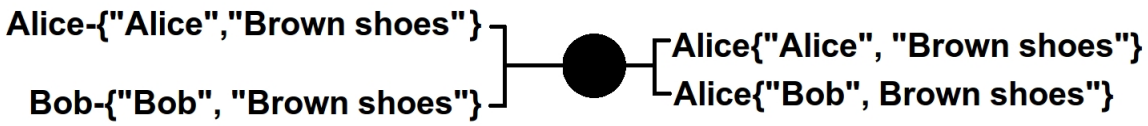

Figure 2.5: The two possible outcomes if we integrate Bob into Alice after the "Red shoes" → "Brown shoes" replacement operation

We consider below a more relevant example, where we integrate two models together, each of which has a reaction. This shows how the order of operations impacts the final model in a biological context.

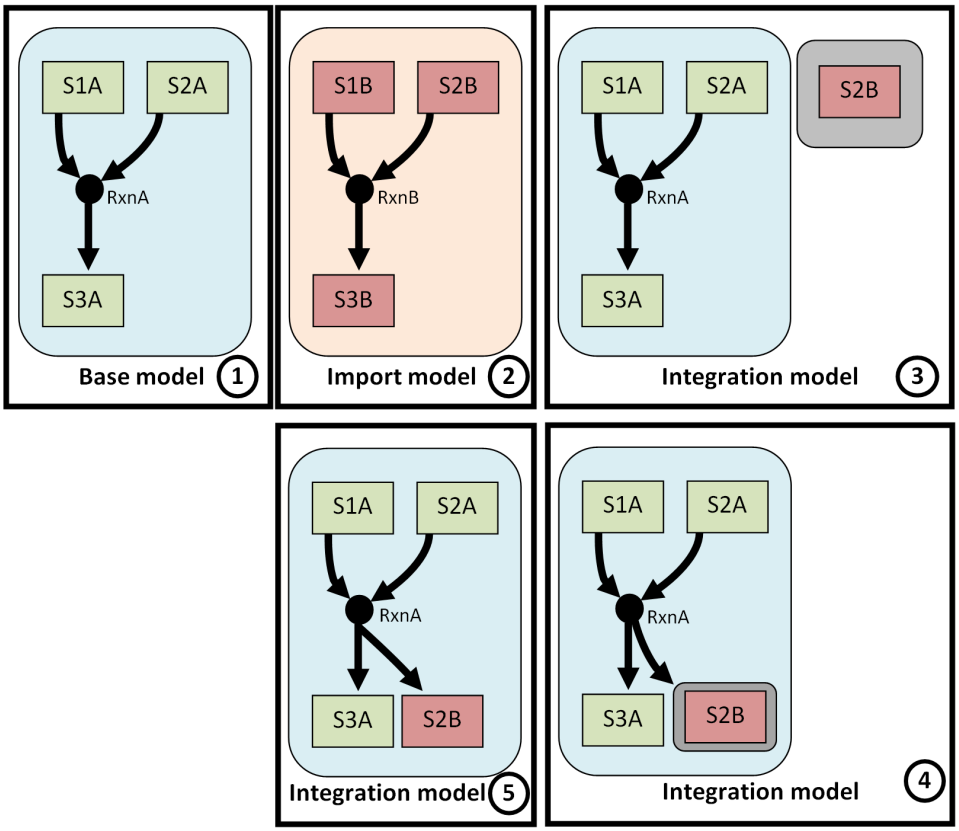

Figure 2.6: We have two models we wish to integrate, model A and model B, where we designate A the **base model** and B the **import model**. We wish to integrate these two models together in the following way **(A)** *Import species S2B into model A*, **(B)** *Integrate RxnA and RxnB, such that the hybrid reaction is [S1A + S2A → S2B + S3A]* and **(C)** *Replace cytosol_B in model A with cytosol_A.*. Figures 1 and 2 show the initial models. Figure 3 shows the integration model after the import of species B. Species B is still in cytosol_B, although in the integration model cytosol_B does not exist, so we represent it's compartment as a grey environment. In figure 4 we integrate the reactions together, which involves introducing S2B as a product. However, it remains in cytosol_B. Figure 5 shows the model after the process of replacing references to cytosol_B with cytosol_A. Nowe S2B is in the correct compartment, and the models have been integrated successfully

As shown, to achieve successful integration the order of operations is critical. If we attempt to integrate the reactions prior to importing the species, this is possible, although for a while the reaction will refer to a species which does not exist. This may be risky, as it entirely relies on the user importing the correct species later. It is this kind of complexity we are attempting to abstract from the user, so this is not ideal. Similarly, if we replace cystosol_B with cytosol_A before the import of S2B, then S2B's "compartment" attribute will still refer to cysto_B. However, cytosol_B does not exist in Model A, and to make the system semantically correct we must re-run the cytosol_B → cytosol_A replacement operation.

This is a very simple example, and in this case it is obvious how to go about this integration in a manner that would achieve the correct result. However, when dealing with two large, complex systems, this process is not trivial. With this in mind, we have defined a specific methodology for the integration of two models.

2.3.2 Import, Replace, Integrate

Below we describe the defined order of operations in IRI integration;

1. A list of all the elements from the import model are imported into the integration model. This occurs exactly once, and is the first thing to happen.

2. A list of all the replacements from the import model are replaced by elements in the base model.

3. Once these two set-up operations have been run, we can begin the interactive integration process. Although not required, the software strongly recommends you proceed the integration in the following order - Unit Definitions, Parameters, Compartments, Species, Reactions, Rules. During the integration process, an element created through an integration operation can be added to the replacement list with the species from the import model as the target to be overwritten - i.e. any reference to the import model's species is corrected to refer to the newly created integration species. Additionally, this new integration element can either replace the pre-existing element from the base model, or can become a new element in its own right.

With the model wide import and replacement operations happening once at the start, we begin the integration operations with the model in as a prepared state as possible. While the import and replacement operations are model wide and automatic, the integration operations are carried out on each of the ten element types individually. It is possible to integrate all ten element types back to back. However, it is also possible to re-run the replacement operation between integration operations. This is typically not necessary, but may provide a means to recover an error made in the integration process.

By forcing the initial import and replacement, but allowing for stepwise integration of different element lists with repeated (but not required) replacement of the up-datable replacement list, there is flexibility in terms of how the integration proceeds, but hopefully not at the expense of maintaining data integrity. This IRI integration also has the advantage of facilitating it's specification through a configuration file. Using such a file, we can define elements to be imported, pairs to be replaced and pairs to be integrated, pre-defining the actions away from the actual software operations. By providing this separation of definition and action, we create a system which can repeatedly integrate together similar models very quickly. In an ideal world a configuration file would not be necessary, and the software would intelligently detect elements which were the same, or seemed relate, to offer replacement or integration options. However, with a total lack of any kind of naming standardisation this is simply not possible[2]. In addition, the flexible design of SBML

[2]Biochemical ontological standardisation is a significant research topic, and it is our hope that it will lead to better, consistent naming conventions

means two syntactically very different models may in fact describe two similar semantic systems. It us up to the user to identify such a scenario, and act accordingly in the design of a configuration file.

2.3.3 Development Language

SBMLIntegrator was developed in C++ for a number of reasons. Arguably the most complete and best documents SBML API is for C++. By starting with a well documented and highly effective API we avoid many of the low level issues associated with data structures, getters, setters and other functionality provided by the API. C++ is also a language ideal for file stream input and manipulation at the character level, as required by the configuration file interface, providing enough low level functionality to construct a complete and fault tolerant file interaction system. Finally, many of the existing SBML tools are already written in C++, so should SBMLIntegrator be absorbed into another, multifaceted tool, this avoids the need for a translation.

2.3.4 Architecture

There are a number of key points relating to the software architecture. The LibSBML API includes an SBML model container class, with container objects for the ten lists described in section 1.4, for each of the elements stored in those lists, and for any other object appropriate attributes associated with those elements. In addition, the API provides getters and setters, as well as functionality to read and write SBML files. Considering this, these predefined container classes were used, with classes containing pointers to Model objects where appropriate. On the whole, however, we've attempted to avoid the issue of class ownership. By treating classes primarily as collections of associated functionality, with significant parent classes holding model objects, we avoid issues of data conflict and duplication.

2.3.5 User Interface

We chose to develop a command line based system, rather than a GUI interface. This was primarily motivated by the restrictive time scale of the software development element of this project. However, as this work has unfolded, there is a noticeable absence of a good GUI SBML model editor for Linux. Command line editors provide an adequate interface for interacting with highly structured data formats such as SBML, however, we have designed the software in such a way that for a final release a QT GUI can be written as an interface over the underlying systems.

For the interface itself, simplicity is a crucial design factor. Command line based systems have the potential to be overwhelming and difficult to understand. One of the main aims for this software was to provide a tool which is easy to use - the difficulties associated with model integration should be limited exclusively to questions relating to the biological significance of decisions, not file format issues, duplication of elements, or clashes between names. Considering this we have adopted a number of standardized approaches to user interaction. These include having three primary ways of interacting with the software which remain consistent through the software.

Where multiple options are available, the user can enter a number from a list to select an option.

```
--------------------------------------
Select components to show
--------------------------------------
[1] ------------ Function Definitions
[2] ------------ Unit Definitions
[3] ------------ Compartments
[4] ------------ Species
[5] ------------ Parameters
[6] ------------ Rules
[7] ------------ Initial Assignments
[8] ------------ Constraints
[9] ------------ Reactions
[10] ----------- Events
[11] ----------- Return
Select:
```

Interface 1: User interaction through a list of options

Where one of two options or one of three options are available a selection of A or B (or C) is presented, rather than a number selection. This is a deliberate separation of the user menu interface, which uses numbers (as shown above in Interface 1) with the interactive integration options (shown in Interface 2, below).

```
### Please select a name for the compartment ###
 A ----- Medium
 B ----- Extracellular_env
 Select (A or B):
```

Interface 2: User interaction where one of two options is available

There are also times where the user must select yes or no to a question. Here, the user is invited to select “Y” or “N”, although “y” or “n” are also acceptable

```
Do you wish to do X?

Please select Y/N:
```

Interface 3: User interaction with a yes or no question

At times the user must enter a number or name - here again we have a similar interface. When the user must (or is advised) to select a element which exists in the model, where appropriate, a list of options is presented.

```
Please enter the new name for the reaction:
Please reselect :
```

Interface 4: Entry screen for value or string

If the user enters an invalid number, a letter or any other kind of illegal character, a standardized error message is generated and they can re-try;

```
**** Invalid selection ****
Please reselect :
```

Interface 5: Typical error message upon invalid input

2.3.6 Configuration File Interface

As mentioned, SBMLIntegrator uses a configuration file to define the elements to import, replace and integrate. The API for this file was developed to be largely separate from the SBMLIntegrator, providing a common interface for the SBMLIntegrator main classes to interact with data structures representing the loaded file in a manner that is totally abstracted from the main functionality, and from any file I/O manipulation.

The SBMLIntegrate configuration file (referred to as the .conf file in the documentation and from here on out) has a regular structure, where for each of the ten element types, we define an import list, a list of paired replacement values and a list of paired integration values. The numbers used in this file refer to the element's index location in the model. We can determine what these values are using the SBMLIntegrator to explore the model and output the list of elements. We suggest configuration file design should be done with the SBMLIntegrator open and being used to determine the correct index values, although this is not required.

```
FunctionDefinitions:
Import: [0,1,2,3]
Replace: [1,2] [2,3]
Integrate: [4,5]

UnitDefinitions:
Import: [1,2,3]
Replace:
Integrate:

Compartments:
Import:
Replace: [2,3]
Integrate:

Species:
Import:
Replace:
Integrate: [4,5]

Parameters:
Import:
Replace:
Integrate:

InitialAssignments:
Import: [1,2]
Replace: [2,3]
Integrate:

Rules:
Import: [3,4]
Replace:
Integrate: [1,5]

Constraints:
Import:
Replace: [1,1]
Integrate: [2,3]

Reactions:
Import:
Replace:
Integrate:

Events:
Import:
Replace:
Integrate:
```

Configuration file 1: Example configuration file structure

The previous page shows an example of a configuration file, giving a general structure. For the paired values (replacements and integration) the first value in the pair comes from the base model, with the second from the import model. In the case of the replacement operation, the base model element replaces any reference to the import model element in the model. In the case of integration operation, the base model element and import model element are combined together.

The .conf file interface uses the keywords Function Definition:, Unit Definition:, Compartment: etc. to act as placeholders for the interface data loader. Once the loader locates these placeholder, it uploads the data following the Import:, Replace: and Integrate: keywords to relevant data structures in the configuration file API. A more detailed discussion of the approach chosen here is provided in the implementation.

A key feature of the configuration file interface is to ensure the state of the data loaded from the file is functional. That is, to provide error and warning messages when a configuration file is badly formatted, and to provide output relating to the data loaded for debugging purposes. Typically, systems which facilitate interactions through a user generated configuration file are exposed to error based on user input through that file, so it is important to ensure that this interface is well guarded against incorrect or pathological input.

2.3.7 Log Files

The integration process is also repeatable, so if the user makes a mistake they can re-do the integration, overwriting previous elements to correct the error. One of the potential drawbacks of a command line interface is that it does not provide any easy mechanism to keep track of a system's state. A GUI interface may provide multiple windows or side boxes, where values exist and are updated in real time to provide an effective at-a-glance evaluation of a systems state. This is not possible with a command line system. With the model integration process, users may not want to be informed of every event going on behind the scenes. Indeed, with complex model integrations, one simple decision such as "*Replace compartment X with compartment Y*" can results in literally hundreds of replacement operations. Consistently outputting all of these to the user would be unhelpful, however, the user may also want some manner to review what operations have taken place, to evaluate a problem with a newly created model and it's source.

With this in mind, for each integration operation the software generates a log file detailing all events and any potential errors or problems it has encountered. In addition to recording all replacement, integration, and import operations, it also details the data loaded of data based on the .conf file, and any problems or errors which occur. For example, if in the configuration file you accidentally defined an import compartment as 20 instead of 2, but there does not exist a compartment 20, then the log file will display the following warning.

```
WARNING - in loading compartments for import from conf file, non-existant
compartment referenced (20). Ignoring...
```

Log file 1: Warning message for undefined compartment

Similarly however, if we had meant to define "[2]" in the import list, but instead defined "[ ]", this causes an error to be written to the log file, and the aborting of the software.

```
ERROR - in conf_preprocess() line 166, two [] back-to-back - this usually
means there is missing data!
```

Log file 2: Error message during configuration file preprocessing

2.4 Methodology

Considering the limited time scale to develop the software, it was essential to determine which components were not relevant to the project and could be left as incomplete stubs. While there is an absolute need to prioritise functionality relevant to this dissertation, this should not be at the expense of the software architecture. We therefore designed initial classes in a complete manner, leaving non-critical functionality with stubs inside completely built functions, containers or classes, rather than simply not doing it.

The overall time scale for the majority of the coding was four weeks, excluding documentation and installation scripts. With the time frame defined, the critical design and development objectives were determined in the context of the overall project's goals. Based on this high level plan with a real completion date, a high-speed scrum like development process was used. Through this, a set of goals were designated and over a sprint period of around three to four days these goals were met. Where there were significant problems, the cause of these problems were evaluated, and the implication considered in the context of the remainder of the project. This is reflected in the implementation of the software. Early design decisions to separate the search and replacement class lead to later decisions to separate the classes which carry out import, replacement and integration. Similarly, despite providing a significant reduction in lines of code, the initial use of a template design pattern in the the cleanup/search classes perhaps made the software more confusing that it needed to be. Subsequently, template classes and functions were increasingly used, while the template design pattern was used less avoiding premature optimisation. This is especially critical considering a significant portion of this base code may need to be restructured for the re-development of SBMLIntegrator into a GUI based tool (not during this project).

2.4.1 Development Tools

Our primary development tool was Emacs, with version control provided by Git. The GitHub repository for this project can be found at https://github.com/rednaxela/SBMLIntegrator. We chose Git and GitHub, as they provide an ideal, decentralized mechanism for version control, which considering a number of different computers were used to carry out development, provided an ideal mechanism not just to maintain regression versions, but additionally to synchronize development across a number of different devices. Testing was carried out through text-dumps, some basic unit testing and user testing. A number of different biochemical systems were used to give a spread of different scenarios for integration, although a more complete testing package would be advisable during beta development.

2.5 Implementation

On the following page, we include the summary inheritance diagram for the SBMLIntegrator software structure. Following this, we will briefly describe each class' role, consider some of the key features in the implementation and the technical decisions which motivated them. We then overview the series of operations which the software performs to take two, separate SBML models and combine them into one single model.

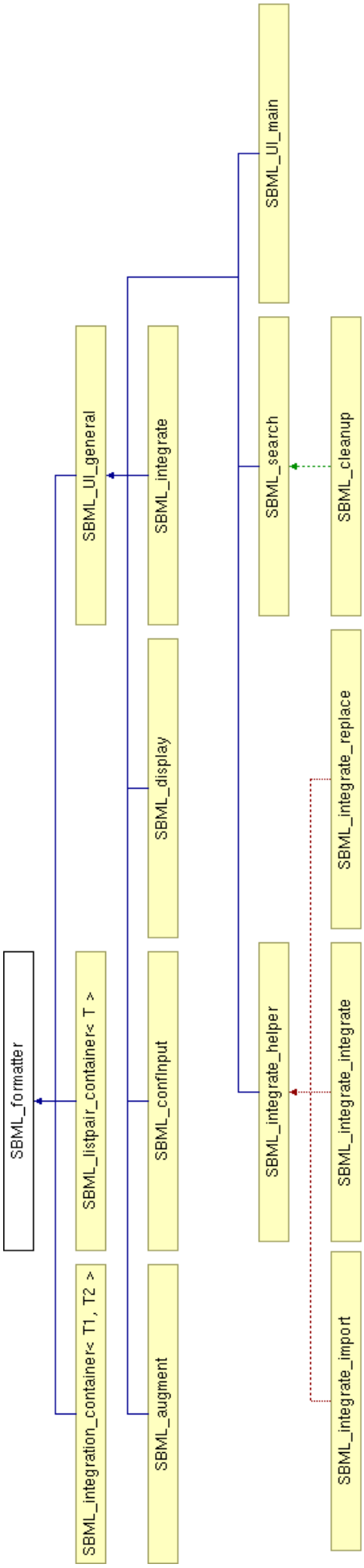

Figure 2.7: This inheritance diagram describes the basic structure of our final implementation of the SBMLIntegrator. On the following page we briefly describe each class and it's primary role

2.5.1 Class Description

The classes are described below in a logical order, although this does not always reflect the order they appear in the preceding diagram. As this is not primarily a software engineering project, we will avoid undue focus on the actual code, although the fully commented source code, complete with separate class and function documentation courtesy of Doxygen[65] is included in the supporting information[3].

SBML_formatter
The SBML_formatter class is the root class to all the other classes in the SBMLIntegrator structure. It contains text and data structure formatting tools as well as storing a static output stream called the log_stream. This log_stream provides a software wide portal to a single log file, through which any operations, warnings and errors can be reported. A number of system-wide constants are set here too, as well as simple functions to get data such as time, date, product version etc. This is also where the template function append_to_ID() exists (discussed later).

SBML_UI_general
Like the SBML_formatter class acts as a root class to everything, the
SBML_UI_general class acts as a root class to anything which has any possibility of user interaction. This encompasses everything except the two container classes SBML_integration_container and SBML_listpair_container. This class contains generic, user interaction functions which may be used by any other class, including functionality such as selecting a value, inputting data or outputting standardized text.

SBML_UI_main
SBML_UI_main provides the core user interface for the main menu, as well as a number of public user interface functions (print logo, print version, print usage etc)

SBML_search
The SBML_search class includes the functionality which allows for the searching through the model for a specific element. This includes functions to determine if a specific element is present (is_present(...)) or not, as well as a unit definition lookup to give the long form of a unit definition. In addition to simply searching for an element, the search framework uses the template design pattern in conjunction with the cleanup class to provide model-wide replacement of an element with an alternative.

SBML_cleanup
The SBML_cleanup class is a child of the SBML_search superclass, and utilizes dynamic binding of methods in the superclass which bind to virtual functions in either the superclass or subclass, depending on what object type is having the method invoked. The cleanup class has a set of public functions for replacing one element with another (replace(element1, element2)). When this method is called it calls the search function locate_and_replace(...). Locate_and_replace is the primary search function in the SBML_search class. However, when called by an SBML_cleanup object it implements a replacement functionality by calling a virtual function which in the search class simply returns true (to indicate the target has been found) but in the cleanup class replaces that element. Through this we effectively use near identical code to achieve radically different results with minimal code duplication.

SBML_augment
SBML_augment is the class responsible for implementing model wide changes. At present, it has a single role in appending a single string to the end of every ID of every element in the model. This is achieved in part through the template function append_to_ID() found in SBML_formatter. This function takes any element and appends a string onto it's ID. It is imperative the user does

[3]Documentation is found the SBMLIntegrator folder under docs. The main page is index.html

not pass an element which does not have an ID attribute to this function.

SBML_confInput
SBML_confInput is a somewhat different class to the ones encountered so far. It is here we implement the configuration file interface. The file interface works by taking a .conf file and scanning it to ensure it has the ten components expected. Once this has been verified, it re-scans through the file, importing the defined values in the import, replace and integrate lists into internal data structures. With the data from the file parsed, the connection to the configuration file is then closed.

The SBML_confInput object has ten internal container objects (SBML_integration_container). Each of these containers can store the import list, two replacement lists (where list_A contains the base elements doing the replacing and list_B the import elements which are to be replaced) and two integration lists (where list_A contains the elements from the base model, and list_B from the import model). With the dual lists, the first item in list A and B are a pair, as are the second, and so on. With the data successfully uploaded from the configuration file, the designated elements are loaded into their appropriate lists in each of the SBML_integration_container objects, and then these are stored permanently as internal data objects in the SBML_confInput object. To access the elements of the lists, getter function, providing access to the container through the SBML_confInput object exist, allowing other classes to request the contents of the container objects, with the SBML_confInput acting as a gatekeeper, maintaining the data's integrity.

SBML_integration_container
As mentioned, this container acts as a data structure to store five lists of elements. It includes getters and setters, as well as a function which returns the length of any of the lists. The paired replace lists A & B and integrate lists A & B should always be the same length.

SBML_integrate
The SBML_integrate class acts as the controller class for the IRI integration process. It contains the user interface menus and houses the interaction which calls upon the classes involved in import, replacement and integration process. It holds a SBML_confInput object with the .conf file data loaded and provides a means for the SBML_integrate_import, SBML_integrate_replace and SBML_integrate_integrate classes to access the relevant information. Additionally, it contains ten SBML_listpair_container objects, which hold the replacement lists, initially copied from an SBML_confInput object. These provide an efficient manner to pass the replacement list between functions, and between classes, and provides a dynamic copy which is updated by the integration process as this proceeds.

SBML_listpair_container
The SBML_listpair_container is a very simple data structure for storing a pair of SBML listOf objects, whatever their element contents may be. It is used to primarily to improve code readability and clarity.

SBML_integrate_import
SBML_integrate_import takes configuration file Import: lists from SBML_integrate and implements the import of the relevant elements from the import model into the integrate model

SBML_integrate_replace
SBML_integrate_replace takes configuration file Replace: lists from SBML_integrate and implements the model-wide replacement of the relevant elements as defined by the two lists.

SBML_integrate_integrate
SBML_integrate_integrate takes the configuration file Integrate: lists from SBML_integrate and implements the model-wide integration of the relevant elements as defined by the two lists. For all

the integrate_*(...) functions, we step through the various attributes of that element type. If two attributes are identical (or not present in both) we skip over them. However, if they differ, we can chose one of the two for our new "integration" element. This is true for all currently implemented integration functions except for reactions. When integrating reactions it is quite probable that neither of the two reactions will obtain the attribute values needed to describe the integrated element. Therefore, where appropriate, the user can define their own value, including kinetic laws, names or reaction constants.

After the integration of an element is complete, the user can select for it to represent either a brand new element, or an update to the existing element in the integration model. Where the integration element can be referenced (i.e. it has an ID variable, such as a species or a parameter, but not a rule) then the user can select to add the newly defined integration element and the import element to as a replacement pair to the replacement list. For example, if they defined a new type of protein based on data from two different species, all future references to the import species would now refer to the newly created species.

We include a very basic UML diagram to describe visually how the preceding classes interact. Note the classes who's role is primarily to provide parent functionality (SBML_search, SBML_formatter, SBML_UI_general SBML_integrate_helper) are not included here for clarity

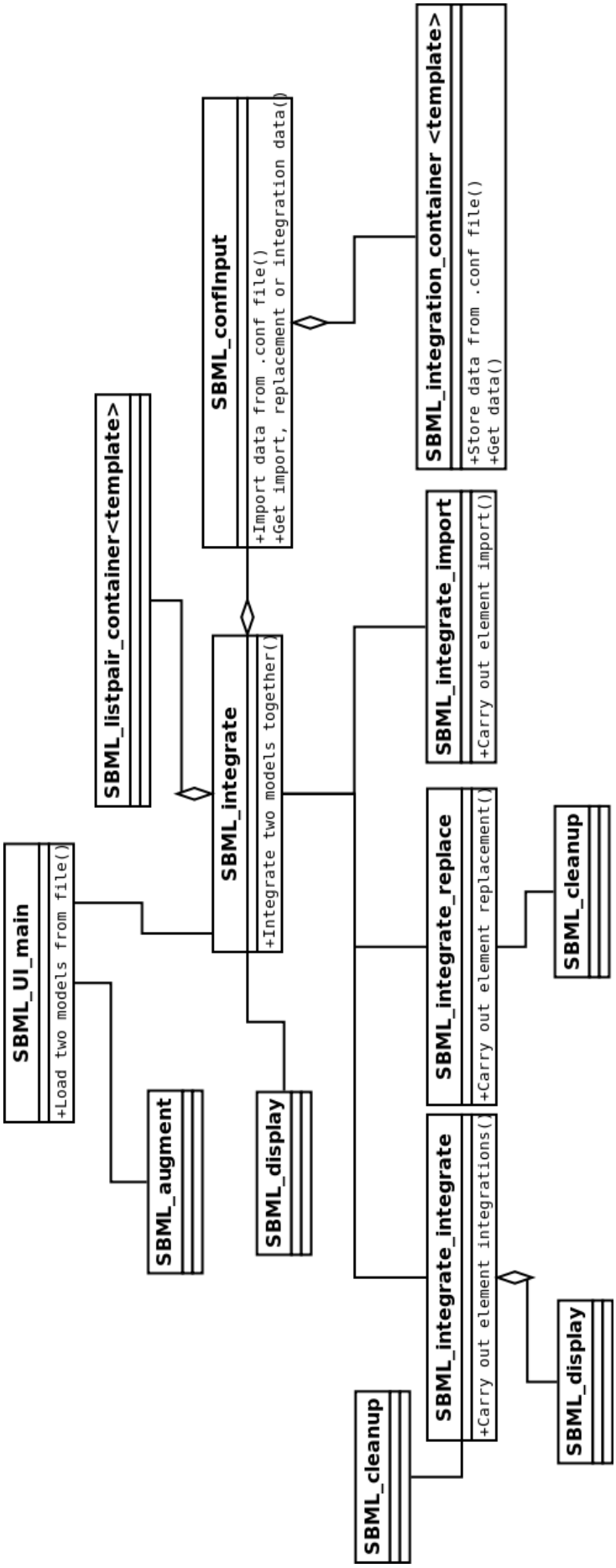

Figure 2.8: Very simple UML showing the software architecture

2.5.2 Functionality Concepts and Implementation

A common theme throughout this development process was the concept of similarity. Frequently, different SBML elements were scanned for a particular string in one or more attributes. Where possible, we tried to implement template functions to take a generic SBML elements as arguments, then use a polymorphic LibSBML function to obtain a value, compare that value to the search target, and act accordingly. However, on occasion specific elements were simply too different from one another to make this approach worthwhile. While our final implementation may not represent the most absolute, code-sparing approach, we have attempted to balance both code clarity and reduction in the form of template functions, with functionality. In some instances, it proved easier to write element-specific functions rather than tailor template functions with specializations.

The use of a template class to store the data from the configuration file enables us to produce a software specific dynamic structure on top of the lower level SBML specific containers implemented by the LibSBML API. Through the template functions we define a standard interface for the main software to interact with the data we import from the two models based on the integration file. The integration file is parsed once, the data used to import all the relative SBML elements from the two models, and then those elementes are stored in a dynamic data structure which can grow and shrink using the underlying SBML ListOf objects. This provides a quick and efficient way to get the relevant data into a stable state. All the disk I/O occurs at the start in a single block, although we chose to carry out multiple reads of the file in favour of a single sequential read, which would require all the data to be in a rigid structure. This represents a significant reduction in disk latency compared to an earlier prototype, where in lieu of storing all the information in memory we carried out individual reads to the file on request. By avoiding the I/O bottleneck we ensure architecturally that the software is optimised, especially for older hardware.

On a earlier prototypes, the ability to select the import and base models on the initial setup was not included. Instead, the software evaluated which model was the bigger and the more complicated, and assigned the larger model as the base model and the smaller as the import model. However, in practice not having control over which is which is not an ideal design feature. It may be the case that the user simply wishes to import a small number of components from a large model into a smaller one. An important design choice is that SBMLIntegrator does not force the user to fully integrate two models. In keeping with SBML's philosophy of not enforcing specific design decisions on model builders, it was decided the forcing the complete integration of two models could reduce usability. Instead, by specifying components the user has the choice of integrating parts of a model. Of course, it is the user's responsibility to ensure they components they import ensure the final model works. However, using the configuration file it is very simple to update the defined model integration parameters should they make a mistake on the first integration.

The use of a detailed logfile began as a debugging tool during development. However, it very quickly became obvious that having an optional detailed output describing the operations the software is carrying out is a valuable tool for model integrators. By being able to see what is happening behind the scenes, much more effective diagnostics, as well as identifying the need to re-run the replacement operations. This information is not critical to the software's functionality, and forcing it upon the user would be unhelpful and overwhelming. However, in equal measure - hiding the complexity to the extent it is not possible to effectively troubleshoot is counter productive. We have therefore opted for a "See as much as you want" approach - the user can easily access the detailed information if they need it, but is not presented with it unless they have a specific need.

2.5.3 Program Operation

We include here a very brief user guide to the software, describing the installation and integration of two models. This is a very brief walk-through, with a more detailed description in the README.txt file included with the source code. As one of the software's objectives, we hope that

the process is

Installation
On this distribution we include the LibSBML API library (used for installation) with the source code, configured to install locally with SBMLIntegrator. In future releases we will offer the option to configure with a pre-existing LibSBML installation, however, for the dissertation hand in it is paramount to ensure we provide a fully, standalone working software. Installation is simply a case of running the installation script as follows

```
COPYING.txt  dxygn_config  INSTALL.sh     README.txt  src
docs         example       libsbml-5.0.0  scripts

$ sh INSTALL.sh
```

Interface 6: SBMLIntegrator installation process

Upon running INSTALL.sh the installation package begins by running the predefined configure process for LibSBML. SBMLIntegrator piggybacks on LibSBML's dependencies, meaning we need only a single configure process for both packages. Should the installation fail during this configure process the user will be greeted with an error message and the library missing. Typically we have tested the installation on a number of systems, and found that the only package missing may be the xml2-config library, an XML parsing library used in favour of the more commonly used Xerces due to an error in Xerces which could not be worked around by LibSBML. This can be obtained by installing the libxml2-dev package.

After the successful configuration process LibSBML is compiled and installed, followed by SBMLIntegator. The installation automatically runs ldconfig, which requires ROOT access. However, if you are unable to enter the root password this is not a major problem, and can be skipped. After SBMLIntegrator has installed you may receive the following error upon attempting to run run the software;

```
$ ./SBMLIntegrator: error while loading shared libraries: libsbml.so.5:
cannot open shared object file: No such file or directory
```

Interface 7: Shared object error

Such an error suggest you need to configure the LD path. In a BASH shell this involves adding

export LD_LIBRARY_PATH=<*full directory where libsbml/lib is located*>:$LD_LIBRARY_PATH

to your .bashrc. Similarly, with csh add the following to your .cshrc file

setenv LD_LIBRARY_PATH <*full directory where libsbml/lib is located*>

In both these cases the libsbml directory is in the folder you ran INSTALL.sh from after installation. No other potential problems have been encountered, and this extra configuration of the LD_LIBRARAY_PATH variable pertain to the LibSBML installation and configuration, not SBMLIntegrator [4].

Usage

With the software installed, it is run using the command;

```
./SBMLIntegrator file [file2]
```

Interface 8: Running SBMLIntegrator, where file and file2 are .xml SBML files

The software takes one or two files. Typical *nix system flags –help and –version appropriately display basic help and the software version. If the program is passed a single file it initiates the software into explore-only mode, meaning it acts as a very simple text-based model viewer.

```
################################################################################
#           WELCOME TO SBML INTEGRATOR (version 0.1 (alpha))                   #
################################################################################

----- SBMLIntegrator - Main Menu -----
   Select an option from below:

 [1] ------------ Explore Models
 [2] ------------ Display Model Summary
 [3] ------------ Quit

 Select:
```

Interface 9: Single file input welcome screen

From here the user can explore a model or display a summary of the models parameters. Figures of these outputs are available in the appendix. To exit the software, the user selects quit. This goes through a process of closing output streams to ensure data integrity of the log file

[4]For more details on these issues see http://sbml.org/Software/libSBML/docs/cpp-api/libsbml-installation.html

If two models are loaded, the user is greeted by the same screen, but with an additional "Integrate models" option. Selection of this triggers the import and initial replacement phases, as discussed previously, leading to the following screen;

```
############################################################
####################   Integrate model   ###################
############################################################

Import model is: MAPK_L2V1.xml
 --> Model ID [Huang1996_MAPK_ultrasens]

Base model is: p38model.xml
 --> Model ID [ ]

 [1] ------------ Integrate Function Definitions (0)
 [2] ------------ Integrate Unit Definitions (0)
 [3] ------------ Integrate Compartments (0)
 [4] ------------ Integrate Species (1)
 [5] ------------ Integrate Parameters (0)
 [6] ------------ Integrate Initial Assignments (0)
 [7] ------------ Integrate Rules (0)
 [8] ------------ Integrate Constraints (0)
 [9] ------------ Integrate Reactions (0)
 [10] ----------- Integrate Events (0)
 [11] ----------- Explore models
 [12] ----------- Explore replacement, import and integration parameters
 [13] ----------- Write integrated model
 [14] ----------- Re-run replacement
 [15] ----------- Return to main menu
 Please select an option:
```

Interface 10: Main integration screen, with two example models loaded. Note the numbers in parenthesis after each of the Integrate <element type> options refers to the number of elements to integrate.

From here, the user can integrates each element type in a stepwise manner. They can re-run the replacement operations (as discussed) and the replacement list can be updated. Full description of the interactive integration process and more can be found in the README.txt file. We do not include it here as it is entirely self explanatory - each step is a question with two or more options and instructions relating to those options. Redundant information is not displayed, and upon integration of an element type a (DONE) message pops up to the left of that element type. Users can re-run an integration step, however, bear in mind this does **not** reset the previous integration, but re-runs the integration between the newly integrated species and the import species, not between the base model species and the import species.

In addition to running the integration, the user can explore any of the three models by selecting Explore models. When the user is satisfied the integration is complete, they can write the integration model to file by selecting Write integrated model. Functionality not mentioned is easy to use, and as mentioned, help messages provide ample instruction regarding all of the interactive integration choices.

Code documentation

In addition to the brief discussion on the software code provided here, a far more in depth overview can be obtained by viewing the source code documentation. We provide a convenient and easily accessible HTML guide created by Doxygen to overview the roles of classes, functions and member attributes. Above are screenshots of the interface, showing the overall class overview, and the structure of the class documentation.

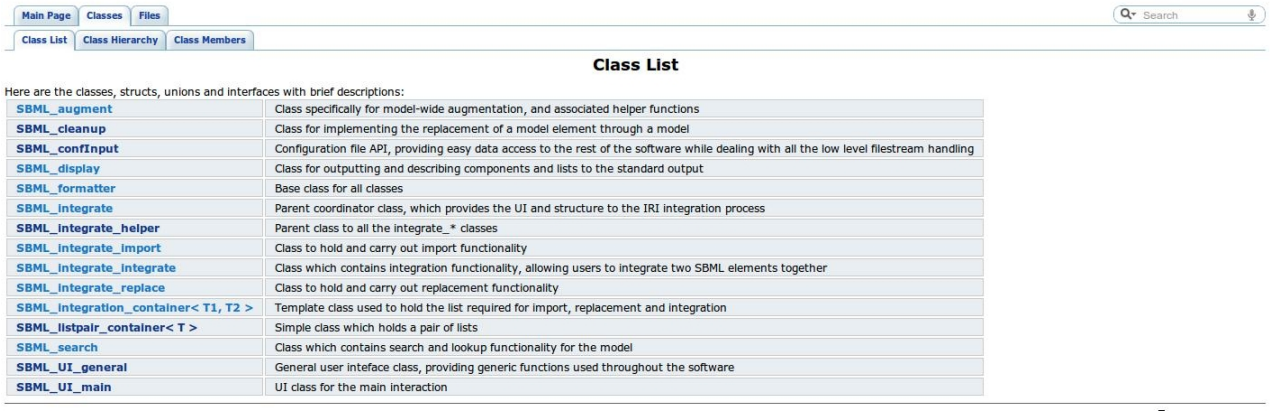

Figure 2.9: Main class list page

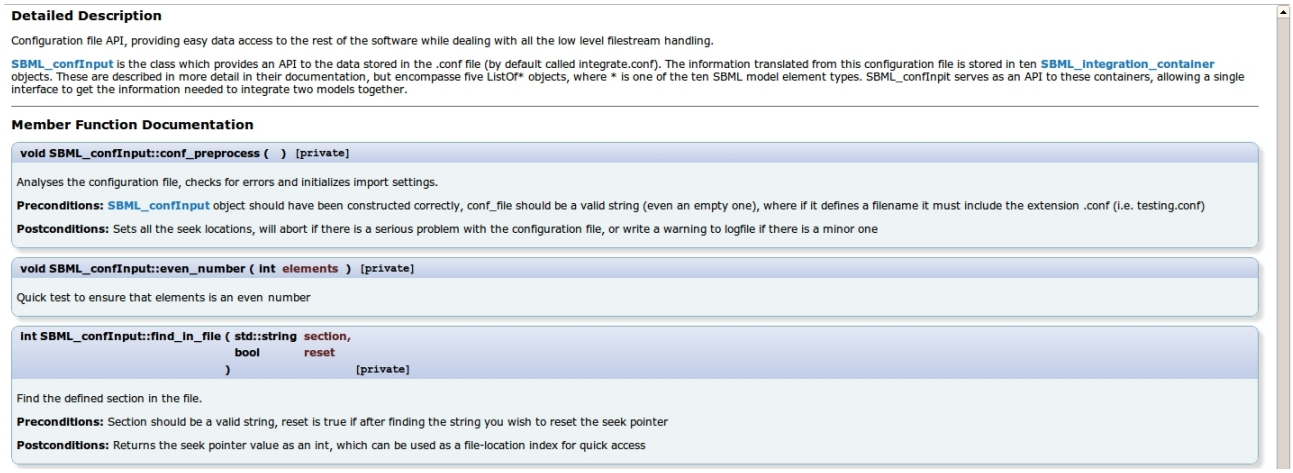

Figure 2.10: Example documentation, here showing the detailed description of the SBML_confInput function

2.6 Evaluation and Future Development

In this section we consider if the SBMLIntegrator in its current form meets our initial objectives. We review the features which were left out in order to accomplish development within the time frame for our specific purpose, and consider what steps were made to ensure their integration in future versions. On this topic, we look ahead to the future of SBMLIntegrator, in terms of future releases and new features added to the software.

2.6.1 Evaluation of Goals

Below we present an overview of the project's goals, and the success with which each was met;

Objective	**Notes**	**Evaluation**
Functionality	Our software is capable of integrating two basic SBML models together. Such models do not include initial assignments, events, constraints or function definitions. However, these elements are more advanced features not necessary for simple systems biology modelling. Integration occurs in a systematic, clear manner, which preserves data integrity while remaining flexible.	Through extensive user testing of model integrations we believe the software more than adequately meets this goals, hugely improving the time it would take to integrate two models by hand.
Ease of use	Wherever necessary, help messages and extra information provide supporting guidance through the software. We include the generation of a log file to facilitate easy debugging if necessary, but avoid an overwhelming amount of information where it is not required	We believe the software is intuitive and easy to use, and that the complexity relating to the integration process has been abstracted to biologically relevant questions exclusively
Ease of installation	Installation requires a single command, with potentially the addition of a file to the LD Path.	We believe the software offers a very straight forward and compelling installation process
Use of a configuration file	A configuration file is used to define the elements to import, replace and integrate. A well defined API for reading the file and accessing the parsed data was developed, which as well as making data access straight forward for the software, provides a number of error checking and defensive programming strategies against mal-formatted files	A configuration file is used
Completed in a timely manner	We deployed a regimented development process to ensure deadlines were met and a product was produced on time. The development of the installation process took considerably longer than expected, however, despite this the version deployed meets our other objective and was completed well before the project's September 9th deadline	The project was successfully completed without becoming an overwhelming burden on the dissertation

Table 2.1: Evaluation of SBMLIntegrator's goals

2.6.2 Future Work and Long Term Goals

The current SBMLIntegrator version does not presently permit the integration of complex systems[5]. These are more advanced modelling features which were not part of our models, and represent functionality and semantics ignored MATLAB. Therefore, had these elements been included in the models there would have been a need to remove them again before they could be simulated. Despite not including functionality relating to the integration of these elements, we have structured the software as if they exist, with the source code including the completed function calls, devoid of content. An early future update to the software will be to include functionality relating to these elements, and allow their integration.

At present, the software can only deal with L2V1 models. While this represents the lowest common denominator, it would be preferable to facilitate multiple levels and versions of support. As a result of the backwards compatibility of SBML, this could be easily facilitated through inheritance, whereby a base class would allow the integration of L2V1, with progressive subclasses allowing for the integration of later SBML versions. Virtual functions and dynamic binding could specify version specific behaviour where different approaches are required depending on the version.

A better user interface is a primary objective for the version 1 release. Through discussion with the SBML community we are identifying desirable features for a simple, fast SBML model viewer, editor and integrator. Adding complete model editing power to SBMLIntegrator is a natural step forwards, although was not one of our initial goals, and considering the time frame was deliberate not attempted in this project. In addition, we are looking at taking the underlying functional core developed here, and implementing a QT framework GUI above it to allow fast and effective user interaction. The command line approach is adequate for this project, and indeed has proven to be necessary to allow us to complete the software in time. However, with more time, a GUI interface has a number of significant advantages over a command line one, and we hope in the months following this project to complete and push out a fully functional GUI based version to meet the need identified during the early phases of this project. Additional discussion on the future work relating to SBMLIntegrator can be found in section 6.2.3.

[5]Where a complex system contains one or more of the following element types: Function Definitions, Initial Assignments, Constraints or Events

Chapter 3

Model Development

In this chapter we describe our development of an integrated p38 MAPK-GC signalling pathway. Although quite a short chapter it was a significant part of the project, as it required a extensive amount of background research. Ultimately, we present an entirely new SBML model, which we hope will in the future help demonstrate the existence of crosstalk between the p38 MAPK and GC signalling pathways.

Early in the project, we held discussions with Professor Ian Adcock, who planned to carry out wet lab experiments to obtain empirical data relating to the species defined in our model. The hope was that by comparing the data generated through Monte Carlo parameter estimation simulations with the real, empirical data, we would be able to identify parameter sets which, upon simulation using our model, would generate results in line with the available experimental data. By comparing how those parameter sets change when the two pathways are integrated (specifically in relation to the p38-MAPK pathway) we hope to identify any impact the GC pathway has on the p38 MAPK pathway and vice-versa, which would be indicative of the crosstalk suggested in the introductory chapter.

3.1 GR Pathway Development

As described in figure 1.8, our starting point was the p38 MAPK model developed by Pfizer. A number of possible related pathways were considered for exploration and development in the context of the p38 MAPK pathway. These included the Interleukin-2 (IL-2), Interleukin-4 (IL-4) and Interleukin-13 (IL-13). Research suggested the IL-13 may be one of the crucial mediators of asthma[71], however upon research into pathway structure it was determined insufficient data exists to generate an effective model. Similarly, despite their potential role, both IL-2 and IL-4 have extensive roles in a number of other processes[52], suggesting that any simplified pathway would have fundamental and potentially overwhelming flaws due to the absence of any interaction with typical cellular components. GCs, however, have a much more specific role, and are almost exclusively associated with the anti-inflammatory response, and it is with this activity that they fulfil their role as a treatment for chronic asthma.

With this in mind, the GC pathway seemed an ideal starting point for designing a complementary signalling pathway which could be merged into the previously designed p38 MAPK pathway model. Developing a new model pathway is not an easy task. There are necessary simplifications which must be made to make the process tractable, and identifying the core pathway components while avoiding unnecessary periphery can be impossible when data is lacking. For this development, an extensive and broad research phase was carried out, looking at the various possible components in the GC pathway. In this development process, there is a need to carefully balance model complexity with completeness; if the model is too simplistic we miss out on key nuances which give characteristic system behaviour. Inevitably a model will be far more simple than the biological

reality. However, with our understanding of a biochemical pathway, there is a fundamental bias towards proteins or pathway components that are well known, easy to isolate, biochemically stable and easy to detect. This means while there may be a significant amount of data for component X, X may not be a core element of this pathway, but instead represents a species for which detection assays are highly sensitive, a species which is more abundant, or a species which is simply implicated by virtue of its role in other similar pathways. There is a need to bear these pitfalls in mind during the model development process.

On the following page we introduce our GR signalling pathway. The schematic provides an overview of the model, followed by a more detailed description of the model. We have numbered the reactions starting at 36 as the reactions in the p38-MAPK pathway go from 1 to 35. Therefore, to facilitate an easier nomenclature in the integrated model, we have begun ours at 36.

Global Reaction Number	**Reaction ID**	**Reaction name**	**Reaction constants**
36	rxnGR_1	GR + GC association	GR_rxn1_kf, GR_rxn1_kr
38	rxnGR_2	GR + p38p association	GR_rxn2_kf, GR_rxn2_kr
39	RxnGR_3	p38P_GR catalysis	GR_rxn3_kc
37	RxnGR_4	GR dimerization	GR_rxn4_kf, GR_rxn4_kr
41	RxnGR_5	JNK activation	GR_rxn5_kf, GR_rxn5_kr
42	RxnGR_6	JNK + AP1 association	GR_rxn6_kf, GR_rxn6_kr
43	RxnGR_7	AP1_JNK catalysis	GR_rxn7_kc
45	RxnGR_8	AP1P + GR association	GR_rxn8_kf, GR_rxn8_kr
46	RxnGR_9	Nf-kB + IkB association	GR_rxn9_kf, GR_rxn9_kr
47	RxnGR_10	Nf-kB_IkB + complex association	GR_rxn10_kf, GR_rxn10_kr
48	RxnGR_11	Nf-kB_IkB_complex catalysis	GR_rxn11_kc
40	RxnGR_12	GRP spontaneous dephos.	GR_rxn12_kf
44	RxnGR_13	AP1P spontaneous dephos.	GR_rxn13_kf
49	RxnGR_14	GR + Nf-kB association	GR_rxn14_kf, GR_rxn14_kr
50	RxnGR_15	GR_dimer nuclear impor	GR_rxn15_kf, GR_rxn15_kr
51	RxnGR_16	NfkB nuclear import	GR_rxn16_kf, GR_rxn16_kr
52	RxnGR_17	AP1P nuclear import	GR_rxn17_kf, GR_rxn16_kr
53	RxnGR_18	GR dimer + antiInf DNA association	GR_rxn18_kf, GR_rxn18_kr
54	RxnGR_19	MKP-1 protein expression	GR_rxn19_kf
55	RxnGR_20	IkB protein expression	GR_rxn20_kf
56	RxnGR_21	Nf-kB + proInf DNA association	GR_rxn21_kf, GR_rxn21_kr
57	RxnGR_22	AP1P + proInf DNA association	GR_rxn22_kf, GR_rxn22_kr
60	RxnGR_23	Pro Inf. DNA + GR dimer association	GR_rxn23_kf, GR_rxn23_kr
58	RxnGR_24	Pro inf. protein expression1	GR_rxn24_kf
59	RxnGR_25	Pro inf. protein expression2	GR_rxn25_kf
61	RxnGR_26	IkB nuclear import	GR_rxn26_kf, GR_rxn26_kr

Table 3.1: GC signalling pathway - Reaction numbers, IDs, names and associated rate constants. Please note that the GR_rxn<number>_k*_ASSIGNED rate constants are in fact not constants, but instead reflect the result of the assignment rule, where the GR_rxn<number>_k*_ASSIGNED is a variable defined by the equation $GR\_rxn < number > \_k * \times compartment\_volume$ where the compartment depends on the direction of the reaction.

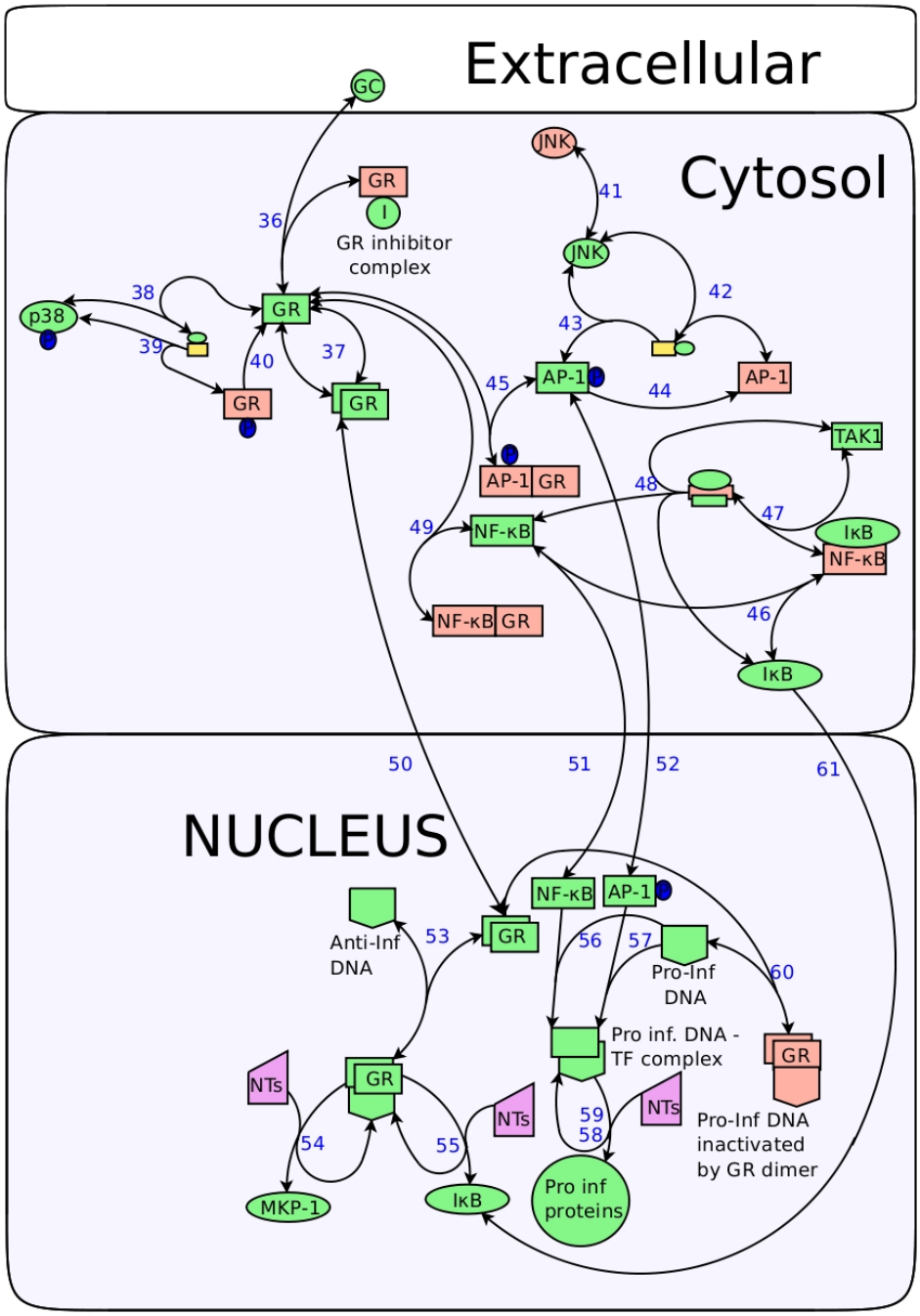

Figure 3.1: **Glucocorticosteroid signalling pathway**. GC binds to the inactive GR-inhibitor complex, causing the release of the inhibitor and activating the GR. The GR monomer can bind to p38P, where p38P then phosphorylates and inactivates the GR. It can also bind the active AP-1P transcription factor, sequestering it and down regulating it's pro-inflammatory impact. Similarly, it also sequesters the NF-κB transcription factor to reduce it's pro-inflammatory effects. Finally, the GR can also dimerize, forming a dimeric transcription factor. This GR dimer undergoes bidirectional transport into the nucleus where it binds to anti inflammatory genes and triggers anti inflammatory protein expression, including the expression of MKP-1, the p38P dephosphorylase and IκB, the NF-κB specific inhibitor. In addition to this positive transcription regulation, GR can also bind to pro inflammatory genes, inhibiting the binding of transcription machinery and blocking their expression. AP-1P, mentioned previously is formed through AP-1 phosphorylation which is catalysed by JNK. AP-1P is dephosphoryalted by a number of proteins, an similarly JNK is activated and deactivated by a number of different factors, this includes both positive and negatively acting phosphorylation and displays extensive tissue specificity. To simplify this, we describe JNK's activation/inactivatation and AP-1Ps dephosphorylation as a simple first order reversible reaction. If AP-1P remains active and is not sequestered by GR it can undergo nuclear transport and bind to pro-inflammatory genes, triggering the expression of generic pro-inflammatory proteins. TAK1 complex can phosphorylate the NF-κB-Iκ complex, releasing NFκB-Iκ from it's inhibitory subunit and allowing it to move into the nucleus, where, like AP-1P, it triggers the expression of generic pro-inflammatory proteins after binding to pro-inflammatory genes. For the process of protein expression, we combine transcription and translation into a single process, treat all nucleotides as an unlimited pool at a fixed concentration, and consider the only factor which impacts gene expression to be the transcription factor in question. This is a gross simplification, but will suffice for an initial model.[6][7][4][5][35][48][53][62][51][41][14][21][10][37][2]

The model is similar to the p38 MAPK model in a number of respects. We deliberately chose to model conversion reactions as a reversible forwards association step, and a second, irreversible catalytic step. For example p38P and the active GR associate together in reaction 38, and this GR_p38P complex can either collapse back to it's constituent reactants, or progress through the catalytic reaction (39) to form the phosphorylated GR (GRP), while the p38P species remains unchanged. It is important to not that these reactions fail to account for molecular mechanisms in any way, nor do they include the affect of modifiers on the reactions.

In addition to the proteins, the GR pathway includes DNA expression regulation. To describe protein expression in such a simplistic system is difficult. The process of DNA expression is highly complicated, and the effect of positive and negative regulation very delicate, complex and multi-faceted. We have attempted here to define a simple approximation for DNA expression, using the same underlying principles used for protein-protein catalysis. For a review of the core components of protein expression see section 1.3.1. We represent the binding of transcription factors such as the GR dimer, NF-κB and AP-1P to DNA as association reactions, similar to the protein-protein ones described previously. However, we can estimate the concentration of anti inflammatory and pro inflammatory DNA to a much more narrow range compared to the ranges of protein species, based on the fact that the number of relevant genes can be far more accurately estimated than the concentration of a protein. The number of genes remains static, and is consistent for all cells in an organism, while protein concentration is dynamic, and varies widely in a cell specific manner.

As the foundation for our range, typical multi-target transcription factors my bind around 10 different genes with common promoter or enhancer regions. From this we can estimate there are around 10 proteins which can have their expression directly triggered through (say) GR dimer binding to DNA (we assume one gene-one protein hypothesis, ignoring RNA processing). We must estimate the number of nucleotides which GR can use to recognize the promoters of the relevant proteins. This is difficult, as it not only depends on the DNA sequence, but also DNA structure. As a rough estimate, we suggest 40 nucleotides either side of the promoter region, plus 8 nucleotides for the promoter itself. Based on this estimate, each of the 10 DNA sequences where GR dimer could bind includes 88 nucleotides, meaning there are 880 possible nucleotides that GR dimer could bind to and correctly associate with the promoter. These are rough estimates, largely based on general trends and not pertaining to a specific target. We need to determine a general range of orders of magnitude to begin the parameter estimation process, and depending on the outcome we can change this range as seems appropriate. Do not consider these numbers hard values.

Using Avogadro's law we can determine the number of moles;

$$N_a = \frac{N}{n} \equiv n = \frac{N}{N_a} \equiv n = \frac{880}{6.023 \times 10^{23}} = 1.4611 \times 10^{-21} mol \tag{3.1}$$

Using the nuclear volume as 1×10^{-13} we can then calculate the concentration of relevant DNA;

$$c = \frac{n}{V} \equiv c = \frac{1.4611 \times 10^{-21}}{1 \times 10^{-13}} = 1.4611 \times 10^{-8} mol/l \tag{3.2}$$

This gives a concentration of around $0.015 \mu mol$, placing it within an equivalent range of of the concentration of protein species, although somewhat lower than the majority of species, as we would expect. To achieve biochemically relevant DNA binding at this concentration, the k_{on} for GR dimer would need to be at the top end of the range of k_{on}, with second order forward rate constants around $1 \times 10^6 M^{-1}$. Indeed, such a k_{on} rate constant would be consistent with a number of empirical studies evaluating promoter association rates[15][59]. Compared to protein species, the concentration of relevant DNA is static, and although this concentration estimation is effectively no more than an educated guess, it does represent a value which can be quantitatively checked and updated easily, and is likely to be far more consistent between cells irrespective of tissue. For simplicity, we assume a concentration one order of magnitude around $0.01 \mu mol$.

The empirically defined association rates of DNA and transcription factor give an estimated range of between 10^4 and 10^9. However, the process of transcription uses a number of molecular clamps and other mechanisms to ensure that once transcription begins the process continues until completion. For our model this is not the case - here this association rate determines the concentration of DNA:transcription factor which in a gross simplification is then converted in a single reaction to transcription factor and expressed protein. Considering this, the ranges for DNA-transcription factor association should higher than the empirical values predict, and should correlate with the rate of protein expression to ensure it is statistically likely for proteins will ever be expressed.

The rate of protein expression can be less accurately estimated based on empirical data, as we condense both transcription and translation into a single, mass action based reaction. Initially we attempted to define a range based on available empirical data. However, it very quickly became clear that a huge number of factors[1] make such an estimation for our simplified and condensed "gene expression" reaction impossible. Instead, we consider the reaction in the context of this model and simulation, as opposed to the reaction in the context of the empirical data associated with the phenomena. Choosing an (empirically) realistic low rate constant would be incorrect, as such rate constants simply describe the rate at which nucleotides are transcribed, or amino acids polymerized. However, for this reaction we are simplifying the DNA and the transcription factor to two, single species, which combine and produce two species. We must therefore give the reaction a rate constant appropriate for a simple chemical reaction, not a the empirical data associated with gene transcription and translation.

With this in mind, we define the DNA-transcription factor association rate as between 10^8 and 10^{11} M^{-1} min^{-1}, meaning the DNA has a very high affinity for the transcription factors. We define the reverse reaction as between 10^-6 and and 10^-1 min^{-1}, reducing the dissociation rate by half that of other reaction ranges. We then define the forward protein expression rate as being between 10^{-4} and 10^0 min^{-1}. This is a lower range than other reactions, meaning that proteins will still be generated (as the DNA-transcription factor is a super-stable complex) but this generation should occur at a significantly reduced rate compared to other reactions. The nucleotide concentration is then also defined with the same range as normal species, except they remain constant through the reaction by setting their boundary conditions boolean attribute to "TRUE". This reflects that, except under exception circumstances, nucleotides do not represent a limiting factor the protein expression, and our model does not include the pathways necessary to generate more nucleotides

A final comment of the values presented here - these simply represent a starting point to narrow the search space. They are educated, intelligent guesses, based on a significant amount of background work and experience. They are a starting point from which we can build a model, and hopefully using empirical data begin to improve and enhance the range and model description. Other rate constant and initial concentration ranges were based on the values used by Hendriks et al[30] to maintain model parity between the two models.

[1]Factors include tissue specificity, species specificity, oxidation state, growth conditions, gene, transcription factor, and post translational modification

Compartment	Name	Lower conc	Upper conc
medium	medium_GC	10^{-3}	10^{2}
cytosol	cytosol_GR_inactive	10^{-3}	10^{2}
cytosol	cytosol_p38P_GR	0	0
cytosol	cytosol_GRP	0	0
cytosol	cytosol_GR	0	0
cytosol	cytosol_GR_dimer	0	0
cytosol	cytosol_JNK_inactive	10^{-3}	10^{2}
cytosol	cytosol_JNK	0	0
cytosol	cytosol_AP1	10^{-3}	10^{2}
cytosol	cytosol_JNK_AP1	0	0
cytosol	cytosol_AP1P	0	0
cytosol	cytosol_NFkB	0	0
cytosol	cytosol_IkB	0	0
cytosol	cytosol_NFkB_IkB	10^{-3}	10^{2}
cytosol	cytosol_NFkB_IkB_complex	0	0
cytosol	cytosol_NFkB_GR	0	0
cytosol	cytosol_p38P	0	0
cytosol	cytosol_AP1P_GR	0	0
cytosol	cytosol_complex	0	0
nucleus	nucleus_MKP1	0	0
nucleus	nucleus_IkB	10^{-3}	10^{2}
nucleus	nucleus_NFkB	10^{-3}	10^{2}
nucleus	nucleus_GR_dimer	0	0
nucleus	nucleus_nucleotides	10^{-3}	10^{2}
nucleus	nucleus_antiInfDNA	10^{-3}	10^{-1}
nucleus	nucleus_antiInfDNA_GR_dimer	0	0
nucleus	nucleus_proInfDNA	10^{-3}	10^{-1}
nucleus	nucleus_proInfDNA_NFkB	0	0
nucleus	nucleus_proInfDNA_AP1P	0	0
nucleus	nucleus_proInfDNA_GR_dimer	0	0
nucleus	nucleus_proInfProteins	0	0

Table 3.2: Initial concentrations. Non zero values originate either from ranges determined by Hendrix et al, or (in the case of DNA) based on our discussion on the previous page.

Name	Description	Rxn	Lower	Upper	Units
GR_rxn1_kf	GR_inactive:GC association rate	1	10^4	10^7	M^{-1} min^{-1}
GR_rxn1_kr	GR_inactive:GC dissociation rate	1	10^{-4}	10^1	min^{-1}
GR_rxn2_kf	GR:p38P association rate	2	10^4	10^7	M^{-1} min^{-1}
GR_rxn2_kr	GR:p38P dissociation rate	2	10^{-4}	10^1	min^{-1}
GR_rxn3_kc	p38P catalytic rate on GR	3	10^{-2}	10^2	min^{-1}
GR_rxn4_kf	GR dimerisation rate	4	10^4	10^7	M^{-1} min^{-1}
GR_rxn4_kr	GR un-dimerisation rate	4	10^{-4}	10^1	min^{-1}
GR_rxn5_kf	JNK activation rate	5	10^{-4}	10^1	min^{-1}
GR_rxn5_kr	JNK inactivation rate	5	10^{-4}	10^1	min^{-1}
GR_rxn6_kf	JNK:AP1 association rate	6	10^4	10^7	M^{-1} min^{-1}
GR_rxn6_kr	JNK:AP1 dissociation rate	6	10^{-4}	10^1	min^{-1}
GR_rxn7_kc	JNK catalytic rate on AP1	7	10^{-2}	10^2	min^{-1}
GR_rxn8_kf	AP1P:GR association rate	8	10^4	10^7	M^{-1} min^{-1}
GR_rxn8_kr	AP1P:GR dissociation rate	8	10^{-4}	10^1	min^{-1}
GR_rxn9_kf	NfkB:IkB association rate	9	10^4	10^7	M^{-1} min^{-1}
GR_rxn9_kr	NfkB:IkB dissociation rate	9	10^{-4}	10^1	min^{-1}
GR_rxn10_kf	TAK1:NFkB_IkB association rate	10	10^4	10^7	M^{-1} min^{-1}
GR_rxn10_kr	TAK1:NFkB_IkB dissociation rate	10	10^{-4}	10^1	min^{-1}
GR_rxn11_kc	TAK1 catalytic rate on NfkB_IkB	11	10^{-2}	10^2	min^{-1}
GR_rxn12_kf	Spontaneous GRP dephosphorylation	12	10^{-4}	10^1	min^{-1}
GR_rxn13_kf	Spontaneous AP1P dephosphorylation	13	10^{-4}	10^1	min^{-1}
GR_rxn14_kf	NfkB:GR association rate	14	10^4	10^7	M^{-1} min^{-1}
GR_rxn14_kr	NfkB:GR dissociation rate	14	10^{-4}	10^1	min^{-1}
GR_rxn15_kf	GR dimer nuclear import rate	15	10^{-4}	10^1	min^{-1}
GR_rxn15_kr	GR dimer nuclear export rate	15	10^{-4}	10^1	min^{-1}
GR_rxn16_kf	NfkB nuclear import rate	16	10^{-4}	10^1	min^{-1}
GR_rxn16_kr	NfkB nuclear export rate	16	10^{-4}	10^1	min^{-1}
GR_rxn17_kf	AP1P nuclear import rate	17	10^{-4}	10^1	min^{-1}
GR_rxn17_kr	AP1P nuclear export rate	17	10^{-4}	10^1	min^{-1}
GR_rxn18_kf	GR dimer:antiInf DNA association rate	18	10^8	10^{11}	M^{-1} min^{-1}
GR_rxn18_kr	GR dimer:antiInf DNA dissociation rate	18	10^{-6}	10^{-1}	min^{-1}
GR_rxn19_kf	MKP-1 protein expression rate	19	10^{-4}	10^0	min^{-1}
GR_rxn20_kf	IkB protein expression rate	20	10^{-4}	10^0	min^{-1}
GR_rxn21_kf	Nf-kB:proInf DNA association rate	21	10^8	10^{11}	M^{-1} min^{-1}
GR_rxn21_kr	Nf-kB:proInf DNA dissociation rate	21	10^{-6}	10^{-1}	min^{-1}
GR_rxn22_kf	AP1P:proInf DNA association rate	22	10^8	10^{11}	M^{-1} min^{-1}
GR_rxn22_kr	AP1P:proInf DNA dissociation rate	22	10^{-6}	10^{-1}	min^{-1}
GR_rxn23_kf	Pro Inf. DNA:GR dimer association	23	10^8	10^{11}	$M^{-1}min^{-1}$
GR_rxn23_kr	Pro Inf. DNA:GR dimer dissociation	23	10^{-6}	10^{-1}	min^{-1}
GR_rxn24_kf	Pro inf. protein expression1 rate	24	10^{-4}	10^0	min^{-1}
GR_rxn25_kf	Pro inf. protein expression2 rate	25	10^{-4}	10^0	min^{-1}
GR_rxn26_kf	IkB nuclear import rate	26	10^{-4}	10^1	min^{-1}
GR_rxn26_kr	IkB nuclear export rate	26	10^{-4}	10^1	min^{-1}

Table 3.3: Reaction constant ranges and units, ranges based on work by Hendriks et al. and our estimations

3.2 p38 Update

The original SBML file provided as supporting information by Hendriks et al. provides an excellent starting point for the p38 MAPK model. It does, however, require some changes to make it usable, as in its current state it fails both to conform the SBML L2V1 specification as well as the requirements which allow it to be simulated in MATLAB. This was a tedious process of removing a significant amount of redundancy, and restructuring the way reaction rates were calculated. In the original p38 model[2], kinetic laws for reactions were defined as a variable, with that variable set by an assignment rule. We reformatted the model so reaction rules were simply the rules themselves, avoiding an unnecessary intermediate assignment.

Reaction constants in the updated p38 model[3] have the format main_P_<type>_<species>. These are the true constants for the reactions. However, the "constants" actually used in the rate equations have the format main_<compartment>_<reaction-number>_<type>, and are in fact not constant. For clarity, we refer to them as applied values. These applied values are instead assigned on each time step through the simulation by an assignment rule, generally of the formula;

$$[main\_ < compartment > \_ < reaction - number > \_ < type > \ = main\_P\_ < type > \_ < species >] \tag{3.3}$$

For the majority of the rules, this introduces unnecessary redundancy. However, it allows us to maintain a single constant, but add additional factors the change the applied value. This is best demonstrated by the nuclear import and export reactions, where the rate is crucially dependent on the nuclear and cytoplasmic volume.

$$[main\_cytosol\_Rxn135\_Kf = main\_P\_ki\_p38P \times comp\_cytosol] \tag{3.4}$$

Additionally, the model provides all units of time as seconds, while their paper uses minutes. To accommodate for this we converted all units which use seconds into minutes by multiplying by 0.016. To ensure consistent units throughout, we converted the LPS concentration from ng/ml into Molar. This is not strictly necessary, however it means the rate constants for the LPS + TLR4 reaction are in equivalent and comparable units to the rest of the model. 10 ng/ml evaluates to 1×10^{-11} or $100nM$ assuming an LPS molecular mass of 100 kDa[8]. This also changed the range of values which the rate constant could be at, although the units stated are describe as $ng/ml/Min^{-1}$, it is our belief that this should in fact read 1/(ng/ml)/Min to give a unit consistent model. As a result, the range is transformed from between 1×10^{-4} and 1×10^{7} to between 1×10^{-5} and 1×10^{6}. Finally, the diagram shown in figure 2.2 is not identical to the one produced by Hendriks et al. We have made a small but crucial adaptation to change reaction 4 from

$$TAK1\_complex\_inactive \rightarrow TLR4 \tag{3.5}$$

to

$$TAK1\_complex\_inactive \rightarrow LPS : TLR4 \tag{3.6}$$

It is the author's opinion that the original diagram contains an error. If the inactive TAK1Complex could degrade TLR4 alone, this would represent an irreversible exit point for LPS from the system. In the original SBML model provided by Hendriks, the reaction uses formula 3.6, not 3.5. We have therefore opted to use this in our model.

[2] p38model.xml in supporting information

[3] p38_idealized_model.xml in supporting information

3.2.1 Testing the p38 MAPK Model

To ensure the restructuring of the p38 model was successfully, we loaded in the best fitting idealized parameter set generated by Hendriks et al. (and used by Baroukh[8]) into the model and ran a 7200 second simulation in MATLAB. The results generated were highly similar to the empirical data, and essentially identical to the JACOBIAN simulations done by Hendriks et al (figures below).

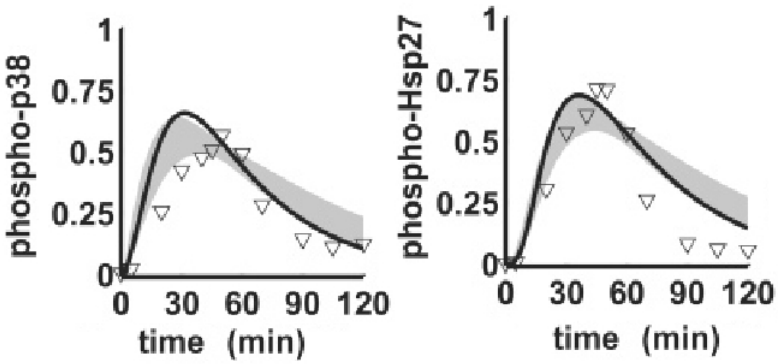

Figure 3.2: 10 ng/ml LPS simulations done in JACOBIAN by Hendriks et al. compared with the experimental data

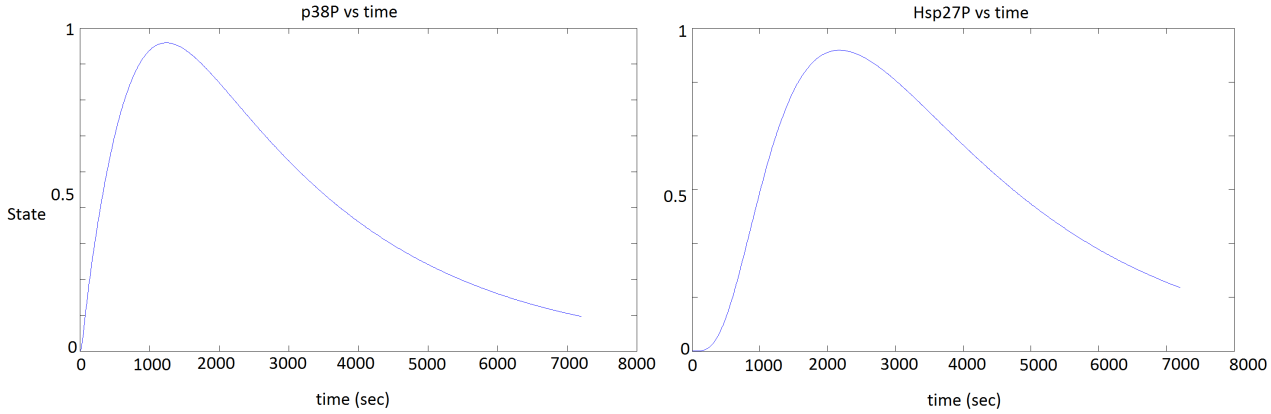

Figure 3.3: 10 ng/ml LPS simulations done in MATLAB

These results were very encouraging. At this stage, it would have also been good to simulate our isolated GC pathway and compared results to the experimental data. However, the required experimental data (do select appropriate parameter sets for the model) was not available at this time.

3.3 Integration Process

With our isolated GR model complete, we used the SBMLIntegrator software described in chapter 2 to integrate our GR model with with the p38 MAPK model. After the significant discussion regarding integration theory in chapter 3 we will not go into detail on how this was accomplished. Instead, we include the integrate.conf file in the supporting information, which details the integration parameters used.

As the GC signalling pathway was constructed with the specific purpose of being integrated with the p38 MAPK pathway, there were no species or reactions to integrate. Designing the configuration file[4] took five minutes, and running the integration was essentially instant. The ease at which these pathways can be re-integrated through the use of the same configuration file (or have their integration parameters updated) allows us to easily make changes and updates to the models individually and then re-run the model integration. This has proven to be invaluable, as we made a number of minor changes and corrections from the original integrated model. We include a full diagrammatic schematic of the integrated model on the following page.

[4]Included in supporting information as integration/integrate.conf

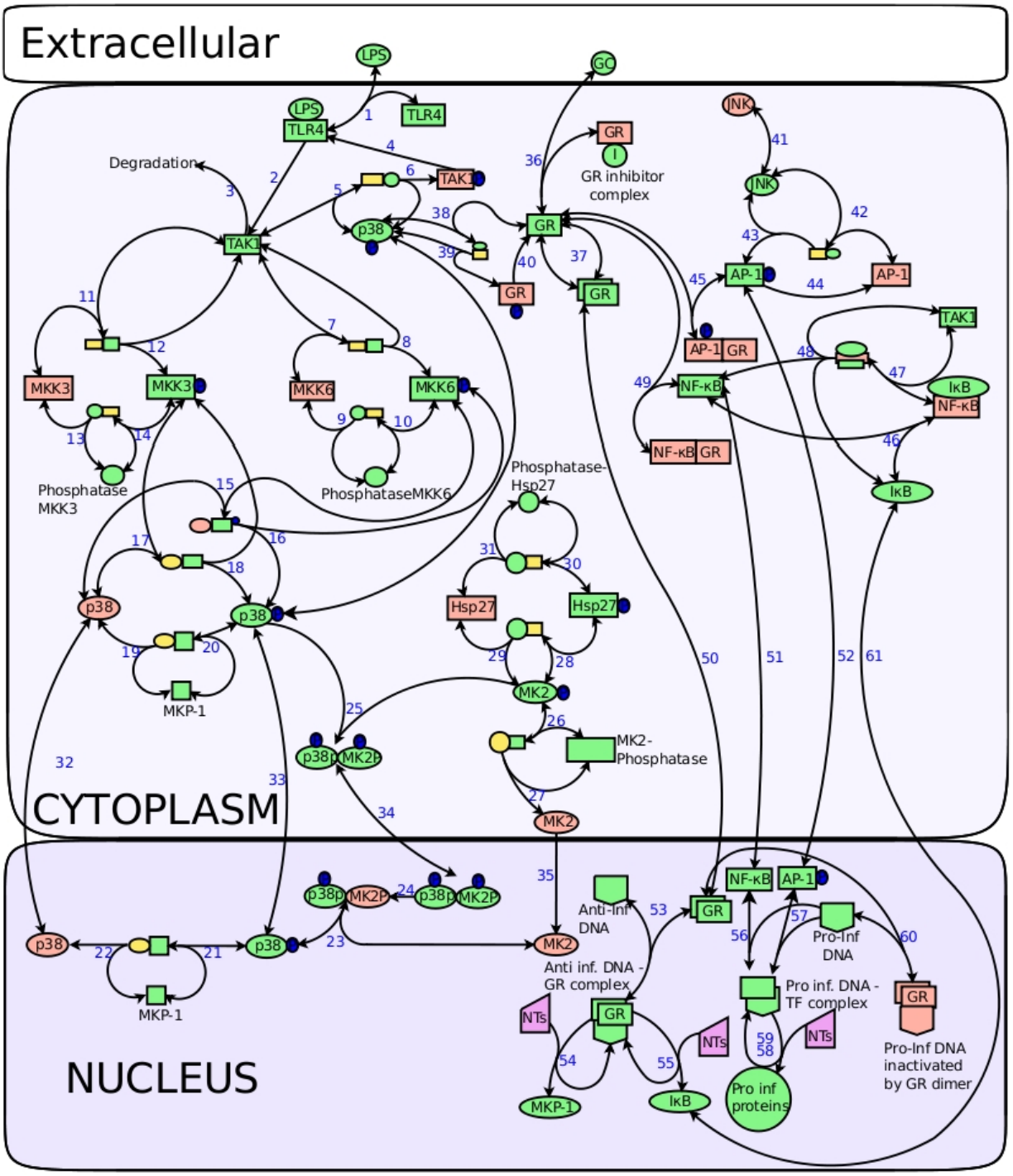

Figure 3.4: **Integrated p38 MAPK - glucocorticosteroid signalling model** - For details of the p38 pathway topology please see the paper by Hendriks et al. For details on the GC topology see figure 3.1

3.4 The Integrated Model

This model is relatively self explanatory, it combines the models defined in figure 1.8 and figure 3.1 into a single model. There are a number of positions along the pathway where crosstalk could occur. p38P provides a crucial integration point between the pathways, and is formed by the action of both MKK3 and MKK6 on p38. p38P binds to active, monomeric GR and phosphorylates it into an inactive form, essentially acting as an off switch for the GC signalling pathway. Similarly p38P binds to and phosphorylates the TAK1 complex into an inactive state, meaning p38P has an antagonistic impact in the two pathways - the TAK1 complex leads to inflammation, while the active GR down regulates it. However, we should consider the species role in the individual pathways to consider the ultimate role of p38P in the integrated pathway. In the p38 pathway, p38P acts to trigger inflammation, and it's role in deactivating TAK1 is one of negative feedback, a regulation control on the pathway to avoid a run-away inflammatory response. In the GC signalling pathway, however, p38P is a pure antagonist to GC's activity, and has no further role.

A second, related interaction point between the two models is that of MKP-1 in the nucleus. Active GR forms dimers which move into the nucleus. Here they trigger the expression of anti inflammatory proteins include MKP-1, the p38P phosphorylase. The increased MKP-1 expression causes an increase in the rate of p38P dephosphorylation into an inactive state in the nucleus, helping to down-regulate it's nuclear activities.

Finally, the TAK1 complex interacts with the IκB - NF-κB complex to release NF-κB. IκB sequesters NF-κB, stopping it from causing pro-inflammatory DNA expression. By freeing NF-κB, the TAK1 complex promotes the expression of of pro-inflammatory proteins indirectly. Active GR binds to and sequesters NF-κB, stopping it from promoting pro inflammatory expression.

Despite having just three points where the pathways interact, they are on three crucial proteins. By comparing the affect of LPS and dexamethasone (a synthetic GC) on the levels of various proteins, we hope to use this pathway to propel the qualitative and quantitative investigation into MAPK-GC crosstalk. To use these pathways, we have developed a set of scripts to carry out Monte Carlo parameter generation and evaluation. By comparing the outcome of simulations run with the data generate by these scripts with real experimental data, we hope to begin to generate putative parameter sets. By evaluating how these sets change in response to different experimental conditions, we hope to gain insight into the signalling process involved.

Chapter 4

Parameter Generation

In this chapter we discuss the final phase of this dissertation, the generation of parameter sets, the simulation of the models, and their evaluation. We consider our design process for the MATLAB scripts which implement these tasks and consider any difficulties and how we overcame them. With the technical description of these scripts complete, the resulting simulation results are described, and the implications of these results considered. It should be noted the primary purpose of this project was not parameter estimation and evaluation. If this were the case, the following chapter would require a far more rigorous approach in terms of the underlying statistics, the mechanism of parameter generation, and the evaluation of the generated results. Instead, this phase was originally part of a means to asses the validity of the integrated model, as well as potentially demonstrate the existence of crosstalk between the two signalling pathways (p38 MAPK and the GC signalling pathway) in the integrated model as described in the preceding chapters. However, the experimental data required to biologically contextualize the integrated model was not obtained, and so as a result crosstalk evaluation was not possible.
Despite this, we include the following discussion, as with the exception of this experimental data, this project is completely prepared to run our Monte Carlo parameter estimation scripts and assess the generated parameter sets. It is our hope that as soon as this data becomes available, this assessment can begin.

We do not include a discussion on the concepts behind the Monte Carlo parameter estimation here, please refer to section 1.5 for further details. Instead, we evaluate our scripts, describing their design and implementation and evaluate their efficiency.

4.1 MATLAB Script Overview

Below we include a structural overview of the driver script which facilitates the process of parameter generation[1]. The script can be seen as two halves, a set up half, were parameters are defined, data is loaded, and the system is prepared for the impending Monte Carlo simulation, and a simulation half. In MATLAB Code 1 we describe, this first part, while the second is described in MATLAB code 2. We use a combination of MATLAB pseudo code and comments to convey the modularity of the procedure, while ensuring the process remains clear and conceptually tractable.

[1] Included in supporting information as /monte_carlo_simulation/additional_information/parameter_estimation_skeleton.m

```
% ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%                                 SETTINGS

%                           Model settings

% Model settings are those relating to the overall model. We set the file
% name for the model and the replacement number.

%                           Species Settings

% Settings defining species related parameters. We set some species numbers
% and concentrations of our variables such as LPS and Dexamethason

%                           Simulation Settings

% Things specific to the ODE simulation and parameter generation steps.
% We define the number of Monte Carlo loop  iterations, the simulation length,
% the max timestep and how we generate random numbers

%                           Ouput Settings

% Output settings are those relating to what data we output, and the system
% we're running on (Windows or Linux)

% ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%                                 LOADING DATA

experimental_data = load_experimental_data(...)

normalized_experimental_data = normalize(experimental_data)

model = load_SBML_model(filename)

% ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%                                 PRE-SIMULATION SETUP

set_formatting_for_windows_or_unix()

% Determine concentration of any species not in Molar (i.e. LPS is in
% ng/ml, so we convert it).

determine_concentrations(...)

% Set simulation settings, including ODE solver to be used (ode23s),
% the relative max tolerance (1e-02) and all the settings you defined
% in the "settings" bit earlier
set_simulation_settings(model)

[missing_initial_concs, missing_params] = identify_missing_parameters(model,
                                          replacement_number)

check_file_structure()
preconstruct_matrices()
```

MATLAB Code 1: Script part 1/2: Setup, loading and initialization

4.1.1 Setup, Loading and Initialization

Settings

The settings provide the input for users using this script to define variables. It should be noted these scripts are very much specific to specific models, although there are some parameters the user may want to alter which are included in the settings section of the script. Below is a summary of those settings and a description of each;

Variable	Description
model_name	Name of the .xml SBML file to be uploaded
replacement_number	Value used by the script to identify parameters which are to be generated
p38P_species_number	Model index number for p38P
Hsp27P_species_number	Model index number for Hsp27P
LPS_species_number	Model index number for LPS
LPS_concentration	Concentration of LPS in ng/ml
LPS_molar_mass	Molar mass of LPS
N_montecarlo_iterations	Number of Monte Carlo loop iterations
simulation_length	Number of simulation seconds to run
max_timestep	Maximum time step through simulation
randomization_type	Type of randomization used (1,2 or 3)
plot_graphs_on_success	Record .fig of P38p and Hsp27p data for working sets
results_folder	Name of folder for output
threshold_val	Sensitivity threshold for good parameters (-1 to 1)
windows	Working in windows (true/false)
unix	Working in a *Nix environment (true/false)

Table 4.1: Settings for MATLAB script

Loading data

Here, three types of data are loaded into the system. The times at which samples were taken as a row vector, one or more sets of experimental results themselves, also as a row vector (where the number of elements in the time points vector must equal the number of elements in the results vector), and the SBML model itself, which is loaded based on the file name defined in settings.

Pre-simulation setup

In this section we carry out all the pre-simulation checks, conversions and set ups. This includes setting the ODE configurations, identifying the number of species in the model and carrying out any unit conversions to ensure parameters are homogeneous in terms of units. We also check to see if a folder exists with the defined folder name - if it does not we create one, so it is imperative the user of the scripts has write permission in the defined output folder. We also use the replacement_number defined in "Settings " to create a matrix which contains the index location of the parameters and initial concentrations which require random values to be generated for them. The variable replacement_number acts as a place-holder to represent, "This value must be generated", which by default we set to 99999. When the user is creating a model to simulate in this script they add this value into any parameter or initial concentration attribute they wish for a random value to be generated. Using the identify_missing_parameters(model, replacement_number) function, we scan through the loaded SBML model, and wherever we find an instance of the value, we index the location of that initial concentration or parameter in the model. This generates two matrices, one for all the indexed initial concentrations (missing_initial_concs) and one for all the indexed parameters (missing_params). We must use a numerical value as this number because the SBML specification ensures that values in parameters or initial concentration must be of type double. However, by hard coding in we run the (somewhat unlikely) risk of replacing a real value which is actually 99999. It is the users responsibility to ensure the replacement_number does not clash with any real values in the model.

```
% ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
%                          MONTE CARLO SIMULATION

for 1 to number of loop iterations
  % Generate a random set of parameters for the model based on the missing
  % initial concentrations and parameters defined in the pre-simulation setup
  [new_conc, new_param] = generate_parameters(missing_initial_concs,
                                              missing_params,
                                              randomization_type)

  % Load newly generate initail_concs and parameters into the model
  model = load_model(new_conc, new_param, model);

  % RUN SIMULATION!
  try
    [t,x,names]= sbiosimulate(model);
  catch error
    keyboard
    continue
  end

  % Normalize all the results
  normalized_results = normalize(results)

  % Compare the results with any experimental data you have
  good_value = compare(normalized_experimental_data, normalized_results)

  % If a parameter set generates good results, save it and the loaded
  % model to results/<parameter_set_number>_<date>
  if (good_value)
     save_data()

  % If not then discard
end of for loop
```

MATLAB Code 2: Script part 2/2: Simulation, evaluation and recording or data

4.1.2 Simulation and Evaluation

MATLAB Code 2 describes the second half of the script, where the parameters are generated, the simulation run and the results compared with our pre-loaded empirical data. The generate_parameters(...) function takes the missing_initial_concs and missing_params variables, and generates random parameters between a predefined range for that specific indexed variable. This means that generate_parameters(...) has the specific ranges and the index location for each possible parameter hard-coded in, and forms the primary basis of why this script is totally model dependent. By hard-coding these ranges and index locations we avoid a massive amount of processing to lookup each value, although as a trade-off we do lose portability in terms of other systems.

In addition to passing the missing_initial_concs and missing_params variables, this function also takes a randomization_type variable, which should be a value between one and three. This provides three different ways to generate random numbers.

- Type 3 is a standard random number generator generating a uniformly distributed value between a defined range. For example, a random number between 1×10^1 and 1×10^4 is

almost always on the order of magnitude of 1×10^3. The problem with this is that when you have significant ranges of values (such as between 1×10^{-5} and 1×10^6 the probability of getting a value less than 1×10^2 is incredibly low. This places an unjustified bias on the upper end of the range of values offered.

- Type 2 is a similar implementation, except it rescales the range to be between 1 and a value, generates a random integer in that range, then rescales that integer back to an appropriate value. For example, if you were to generate a random number between 1×10^7 and 1×10^8, it rescales this range to be between 1 and 10, generates a random int between 1 and 10 (say 4) and rescales this back to be 4×10^7. This does not change the distribution of numbers, but it does limit the number of possible numbers.

- Type 3 is a different implementation which introduces it's own bias but gives a much broader range of values. Here, the range of exponents is determined, and a random integer within that range determines the exponent of the returned value. We then generate a random number between 0.1 and 0.999999, which is multiplied by 10 to the power of the new exponent. For a range between 1×10^{-5} and 1×10^6, this means there is an equal probability of getting a value on the order of magnitude of 1×10^{-5} and 1×10^5. The disadvantage here is that this biases our average value towards the lower end of the range, as in reality there are "more" numbers between 1000 and 10000 then there are between 1 and 10. However, for the sake of generating a wide range of values for a simple randomization function in this context it performs fine, and for that reason the default randomization_type is 3. When types 1 or 2 are used the types of results are very similar on every parameter set generation, however, a much wider range of results (including working sets) are generated by type 3.

Once we have created our matrix of new values the load_model(new_conc, new_param, model) function loads these new values into the model at their correct locations. With this complete we use the aptly titled simulate(model) function to run the ODE simulation using MATLAB's built in ODE simulator. The default simulator used is the ODE23s - chemical reactions are notorious for providing stiff ODEs[61], and our system can afford to use crude error tolerance - there is not a specific correct value to determine. ODE23s is ideal for such a system, and although other engines could be used we would highly recommend against it. The simulated(model) function is enclosed in a try/catch block for easy access if a parameter set causes the system to stall. For example, if the parameter set triggers matrix convergent errors or the system stiffness prohibits progress, suggesting the parameter set is not valid, the user can press [CTRL + C] to abort the simulation. This triggers an exception, meaning we jump into the catch block and invoke the keyboard command, transferring control to the user. We could then select enter "*resume*", which causes us to jump to the next loop iteration or "*dbquit*" which aborts the entire script. Without the keyboard method call we would be unable to quit the simulation midway through without forcing MATLAB to quit.

The normalize_results(results) rescales the results into a normalized form, and then compare(normalized_experimental_data, normalized_results) uses a simple combination of interpolation from the simulation data to the experimental time points followed by a coefficient cost function to determine how similar the results are. If the results are above a certain threshold then we consider this parameter set to be acceptable. The data associated with this run (the parameter set, model, coefficient values for all comparisons, and graphs of those comparisons) are saved, and then process repeats, generating a new random set of values and running the simulation with those newly created values loaded into the model.

4.2 Parameter Generation Ranges

The ranges for our parameters originate from both the original p38 MAPK model and from our own work as discussed in tables 3.2 and 3.3 (GC model) and appendix B1 and B2. These ranges are used as input to the randomizer function in the generate_parameters2.m file. For the rate constant

values, on each loop iteration, a random values is generated which lies within these ranges. For the initial concentrations, the same is done, except only for the species which have a non-zero concentration to begin with (as described by the aforementioned tables and appendices).

4.3 Simulations Run

We ran simulations of p38 MAPK pathway at 1, 10 and 100 ng/ml LPS. Simulations used a threshold value of 0.88, and ran 50 000 Monte Carlo iterations, with a max time step of 100s, leading approximately 72-100 data points per variable per simulation. We present our results of these simulations in the following section. The ability to select parameter sets depends entirely on the availability of experimental data. We had initially expected to obtain data relating to the pathways under the following experimental conditions;

LPS ng/ml	**Dexamethasone**	**MAPK inhibitor**
1	Absent	Absent
10	Absent	Absent
0	Present	Absent
1	Present	Absent
10	Present	Absent
1	Absent	Present
10	Absent	Present
0	Present	Present
1	Present	Present
10	Present	Present

Table 4.2: Varying experimental conditions

These experiments were originally to be run and the results collected in good time (July), but unfortunately various problems meant that this data was not able to be obtained. This was unfortunate - without this data were were unable to run physiologically contextualized simulations of our integrated pathway. However, using the experimental data generated by previously by Hendriks et al. we were able to run the p38-MAPK simulation to ensure our new p38 MAPK model was valid.

In addition, we used our parameter generation algorithm to generate all of the relevant parameters and concentrations for our integrated system, and using the original p38 data ran a number of simulations of the integrated system to check it's biological validity. Such a system is not physiologically relevant - the p38 data is for experimental results where no GC were present, while our integrated system includes GC . However, despite not being physiologically relevant, running the integrated system like this has two major advantages. We are able to confirm that our integrated model runs, that it has been correctly implemented and does not consistently crash, hugely disrupt the p38 results, or give wildly unexpected results. Secondly, assuming the system works correctly, we can check that it is behaving as expected by examining the concentration of inflammatory protein over time. Based on our models, we would expect the level of pro inflammatory proteins to increase - there is no mechanism for their reduction, although they do not interfere with the reaction in anyway, acting essentially as biomarkers for the model's progress. It is predicted that if the integrated pathway is proceeding correctly, the concentration of pro inflammatory proteins will increase (at different but steady rates) through the course of the simulation.

Chapter 5

Results and discussion

We include here some of the data generated by our simulations. All the parameter sets, the complete models and a range of other data is provided in the supporting information. Because of the lack of experimental data and the overall aims of this project, the discussion of these parameter sets is brief. The rate constant values are discussed exclusively, as is their impact on the concentration trajectory (that is, how a species' concentration varies across the simulation), while the initial concentration values do not at this stage add any significant discussion. Some of the implications of these results are considered, although any detailed statistical analysis is avoided. The purpose of these simulations is not the generate parameter sets for analysis (although another project would no doubt find a wealth of information of interest here) but instead as qualitative tools to test the validity of both the re-structured p38 model, and our integrated p38-MAPK model. The results should be biochemically relevant, match previous work, and meet our expectation defined in the previous section.

5.1 p38 MAPK Simulations

Three p38 MAPK simulations were run, with critical simulation parameters defined below;

Simulation	**[LPS] (ng/ml)**	**# parameter sets**	**# loop iterations**	**Corr. threshold**
1	1	53	50000	0.88
2	10	95	50000	0.88
3	100	240	50000	0.88

Table 5.1: Summary of p38 MAPK simulations

A subsection of the concentration graphs produced by the p38 MAPK simulations are included. These graphs show the concentrations of specific species over time generated by every parameter sets which was determined functional based on existing experimental data. These concentration spectrum graphs provide a qualitative overview of the progression of various species over all the operational parameter sets. The objective here is not to analyse these data individually, but instead to use the graphs as qualitative tools from which we can gain insight into the model, the variability on each of those species' and ensure the progressions meet our expectations.

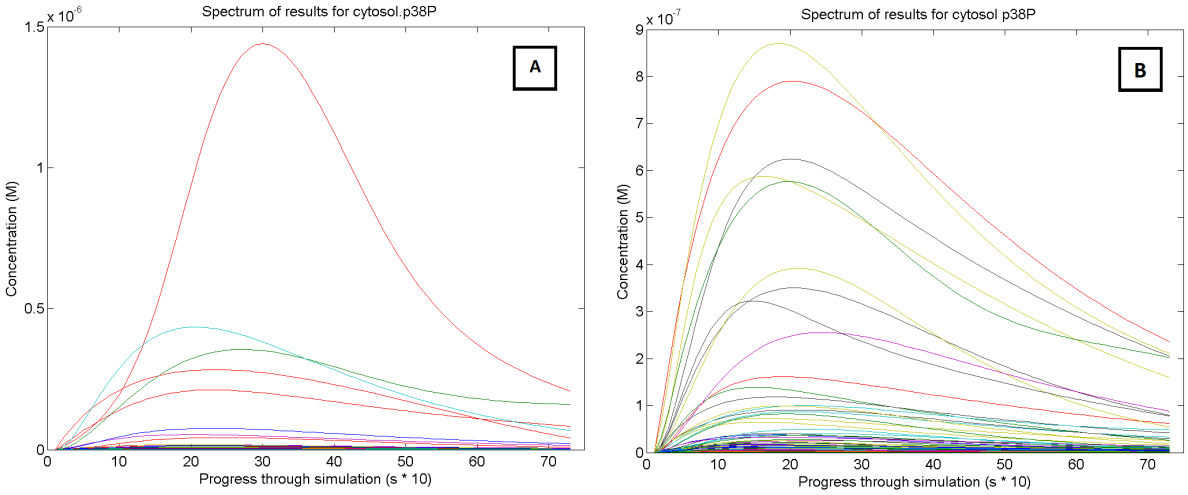

Figure 5.1: **p38P concentration spectrum**. Plot **(A)** describes the concentration at 1 ng/ml LPS, while plot **(B)** is at 100 ng/ml LPS

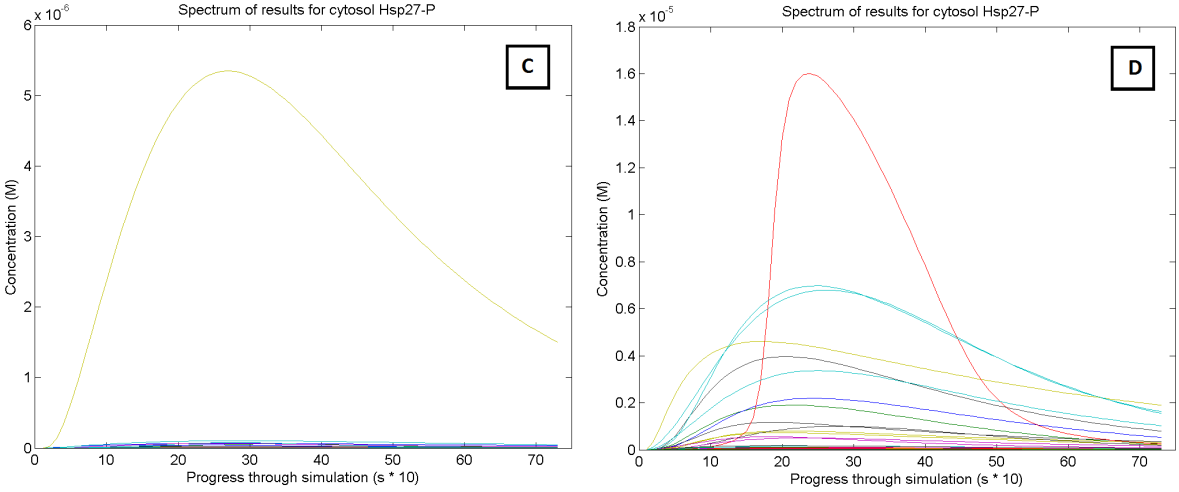

Figure 5.2: **Hsp27P concentration spectrum**. Plot **(A)** describes the concentration at 1 ng/ml LPS, while plot **(B)** is at 100 ng/ml LPS

Despite some variation in absolute levels of both p38P and Hsp27P, there is a consistent shape, which is highly similar to the p38P experimental data. This shape can be seen in the normalized graphs (not shown, but in supporting information). This is in contrast to the levels of the TAK1 complex, shown on the following page. The cytosol degradation and TAK1:MKK3 spectrum graphs (Appendix C.1-A and C.1-B) show very similar shapes, although the ranges vary significantly owing to the different starting concentrations. The MKK6P spectrum (C.2-C) on the other hand, appears to show a number of different behaviours. For example, the prominent red and blue lines have very different trajectories over the simulation's time course. However, they both represent functional parameter sets. C.3-D and C.3-E display similar trends. In all our parameter sets for the p38P graphs, concentration trajectories displayed generally smooth curves (with the exception of the first few time steps where the system reaches an initial equilibrium). This is consistent both with previous work [30][8] and represents a typical species concentration progression as shown in wide range of system biology simulations.

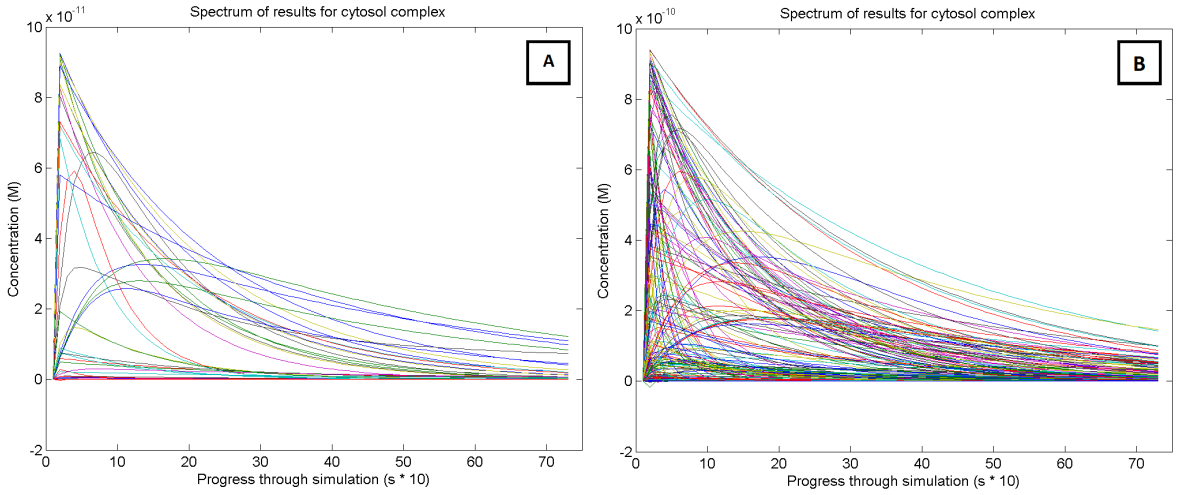

Figure 5.3: **TAK1 concentration spectrum**. Plot **(A)** describes the concentration at 1 ng/ml LPS, while plot **(B)** is at 100 ng/ml LPS

Based on the TAK1 data shown above, it appear that we have a wide range of different progressions. The overwhelming trend is an initial increase, followed by a drop in the concentration of TAK1 (referred to through the graphs as "complex"). However, 5.3-B especially shows a huge range of different paths - some trajectories initial jump very rapidly and immediately begin to decrease in concentration, while others rise for the first quarter of the simulation, then begin to decrease again. This is consistent with a species with a number of different associated reactions (TAK1 can bind to MKK3, MKK6, p38P or undergo degradation), where each of the parameters associated with these reactions is variable.

The table on the following page gives some simple statistics associated with the data generated.

	Mean			STD/mean		
Names	**1ng/ml**	**10ng/ml**	**100ng/ml**	**1ng/ml**	**10ng/ml**	**100ng/ml**
kr_p38PMK2P	2.25	1.71	1.81	1.28	1.73	1.60
kr_LPS_TLR4	1.12	1.35	1.19	2.34	1.99	1.90
kr_p38_MKK6	1.14	1.53	1.01	2.02	1.63	2.09
kr_p38P_Ppase	1.86	1.39	1.52	1.55	1.95	1.72
ki_p38PMK2P	0.61	0.81	0.70	2.43	2.43	2.84
ki_MK2	0.80	1.30	1.17	2.64	2.01	1.99
kr_complex_p38P	1.00	1.14	1.48	1.92	2.05	1.82
kr_Hsp27P_Ppase	0.59	1.20	1.36	2.66	2.17	1.96
ki_p38	1.06	1.49	1.50	2.13	1.74	1.78
kc_p38ppase	11.44	12.65	15.42	1.88	1.93	1.70
kf_MK2_p3	2.29×10^6	2.27×10^6	2.03×10^6	1.37	1.34	1.42
kf_complex_p38P	1.26×10^6	1.63×10^6	1.66×10^6	1.72	1.63	1.56
kr_MKK3_complex	0.83	1.50	1.25	2.43	1.75	1.99
kc_MK2	20.79	25.86	19.58	1.42	1.24	1.41
kf_Hsp27P_Ppase	4.00×10^6	2.13×10^6	2.15×10^6	0.93	1.38	1.31
kf_MKK6_complex	1.50×10^6	1.66×10^6	1.97×10^6	1.76	1.56	1.45
kr_MKK6_complex	1.45	1.84	1.32	1.70	1.66	1.96
kc_Hsp27ppase	8.97	12.33	15.64	2.30	1.88	1.69
k_inactivation	0.79	0.69	0.85	2.61	2.63	2.36
kr_Hsp27_MK2	0.94	1.68	1.19	2.25	1.79	2.00
kf_p38_MKK6	1.51×10^6	2.00×10^6	2.21×10^6	1.74	1.46	1.35
kf_MK2P_Ppase	3.28×10^6	3.18×10^6	2.33×10^6	1.00	1.05	1.28
kc_MKK6ppas	11.49	15.91	14.18	1.82	1.72	1.83
kf_p38_MKK3	2.42×10^6	2.67×10^6	2.11×10^6	1.25	1.26	1.43
kr_MKK6P_Ppase	2.14	1.43	1.44	1.46	1.77	1.88
kc_MKK3ppase	16.76	12.88	14.54	1.68	1.81	1.79
ko_p38P	1.17	1.47	1.39	2.06	1.82	1.87
kf_MKK3P_Ppase	2.06×10^6	1.82×10^6	1.93×10^6	1.43	1.65	1.49
kc_MKK3	14.50	14.37	17.02	1.82	1.93	1.68
kr_MKK3P_Ppase	1.07	1.29	1.19	2.13	1.91	1.91
kc_p38Pcomplex	16.56	17.40	15.32	1.76	1.69	1.74
kc_MKK6	10.75	16.18	12.22	2.15	1.65	1.97
kf_p38P_Ppase	1.02×10^6	1.21×10^6	1.50×10^6	2.03	1.83	1.68
kc_complexMKK3	11.27	13.87	16.44	2.04	1.87	1.73
kf_MKK3_complex	1.79×10^6	1.63×10^6	1.86×10^6	1.48	1.47	1.51
kc_MK2ppase	21.11	19.79	16.94	1.40	1.57	1.57
kf_Hsp27_MK	1.34×10^6	1.82×10^6	1.61×10^6	1.76	1.49	1.57
kf_MKK6P_Ppase	1.68×10^6	1.87×10^6	1.84×10^6	1.65	1.41	1.56
ko_p38PMK2P	2.16	2.12	2.11	1.35	1.41	1.46
k_deg_complex	1.11	0.98	0.71	2.26	2.30	2.58
kr_MK2P_Ppase	0.74	0.96	0.79	2.40	2.43	2.44
k_activation	8.19	5.49	5.13	2.79	2.81	3.22
kf_p38P_Ppase_nucleus	2.51×10^6	2.23×10^6	1.76×10^6	1.29	1.35	1.54
kc_p38	12.18	17.86	13.60	1.97	1.73	1.80
kc_complexMKK6	18.81	15.95	17.13	1.49	1.65	1.76
ko_p38	1.47	1.36	1.30	1.91	1.91	1.89
kr_MK2_p38	1.20	1.12	1.11	2.11	2.18	2.19
kr_p38_MKK3	1.50	1.13	1.18	1.82	2.16	2.01
k_reactivation	0.87	1.30	1.40	2.15	2.03	1.83
kf_LPS_TLR4	1.01E+17	8.46E+16	7.45E+16	2.31	2.52	2.80
ko_MK2	0.93	1.10	0.99	2.17	2.06	2.09
kc_p38P_Ppase_nucleus	11.16	14.80	16.28	2.00	1.77	1.67
kr_p38P_Ppase_nucleu	0.85	1.00	1.34	2.77	2.28	1.97
ki_p38P	1.40	1.33	1.10	1.83	2.06	2.10

Table 5.2: Mean and standard deviation/mean for rate constant values generate through parameter estimation

We will not consider the significance of these parameters. Indeed, the method of random number generation introduces statistical bias which means carrying out analysis of parameter distribution assuming a uniform distribution of random sets would not be appropriate. The mean provides a standard metric with which to asses the average value. Because of the difference in some of the units of these parameters the standard deviation/mean metric is also included, as it gives an easily comparable way of looking at the spread of values around the mean. As shown, these values are fairly similar for all parameters across all LPS concentrations, and indicate a significant spread around the mean. We avoid further statistical analysis, partly because of the nature of the project, but additionally because with such variable data three simulations does not represent enough data to carry out effective data mining. We include a subset of the concentration spectrum graphs in the appendix, with captions describing features of interest. Displaying a selection of the spectrum graphs as opposed to all of them is simply an issue of space, but all forty graphs at the three LPS concentrations, all the parameter sets and a range of other information is included in the supporting information.

Based on our data we feel confident that both MATLAB and our reconstruction of the p38 MAPK model produce results consistent both with the simulation work done by Hendriks et al.[30] and the supporting experimental work. This in itself is a major achievement, as it facilitates a significant potential for investigation into the MAPK-p38 pathway. MATLAB is a universally used mathematical analysis and modelling tool, while JACOBIAN is much less widely used or available. We welcome further research from computational biologists interested in pursuing work on this pathway with more advanced parameter generation and evaluation methodologies.

5.2 Integrated Pathway Simulations

In addition to the p38 MAPK model, the parameter estimation and evaluation scripts were adapted to operate on the integrated model, using the additional parameter and concentration ranges defined in 3.2 and 3.3. As discussed, these simulation are not physiologically relevant, so analysis of the resulting systems is inappropriate. Our metric for a functional parameter set is based upon empirical data where GC are not present, meaning raw values and specific system behaviour is not relevant. However, these simulations tell us a great deal about the integrated model from an operational standpoint.

We ran a single simulation with 10 ng/ml LPS, a threshold value of 0.88 and 50000 loop iterations. This simulation generated results consistent with our previous simulations in terms of scale and concentration progression of species present in both, producing 113 functional parameter sets, while the p38 MAPK pathway at the same LPS concentration generated 95 parameter sets.

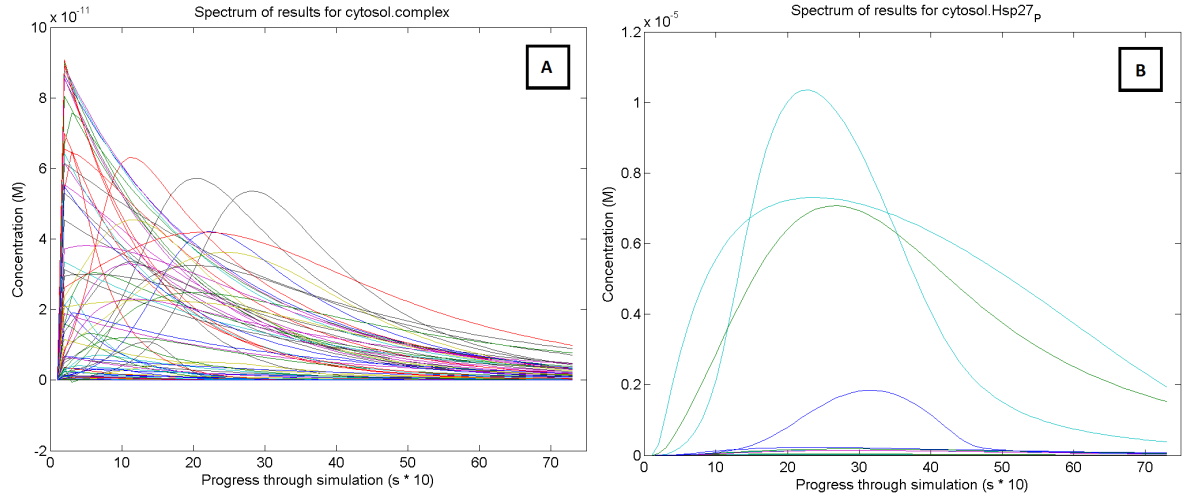

Figure 5.4: Concentration spectrum graphs for **(A)** TAK1 and **(B)** Hsp27P

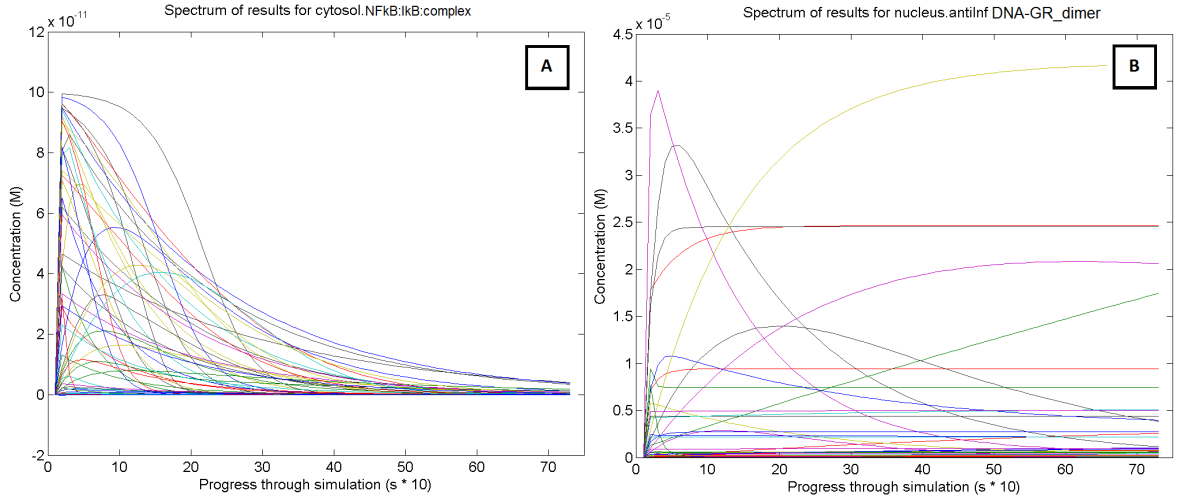

Figure 5.5: Concentration spectrum graphs for **(A)** NF-κB:IκB:TAK1 complex and **(B)** Anti inflammatory DNA:GR dimer

The Hsp27P spectrum is similar to our p38 MAPK simulation spectrum. The TAK1 spectrum is also similar the p38 MAPK TAK1 spectrum, although we do see a number of later peaking tra-

jectories, suggesting there may be additional routes for the TAK1, such as binding the NF-κB-IκB complex. The NF-κB:IκB:TAK1 concentration spectrum has a range of values not dissimilar to the isolated TAK1 data. The anti-inflammatory DNA:GR dimer spectrum contains an even wider range of different trajectories. Some peak very early, decline rapidly and remain there, while others rise slowly then fall, rise over the entire course of the simulation, or even rise in a linear manner. This probably reflects the multi-route nature of GR and the GR dimer - if we consider its position in the integrated pathway GR_dimer can be converted back to GR easily, which interacts with a wide number of species.

The most interesting graph, however, is that of the pro inflammatory proteins, shown below, clearly demonstrating the predicted behaviour of gradual increase of the simulation time-scale at a variety of different rates.

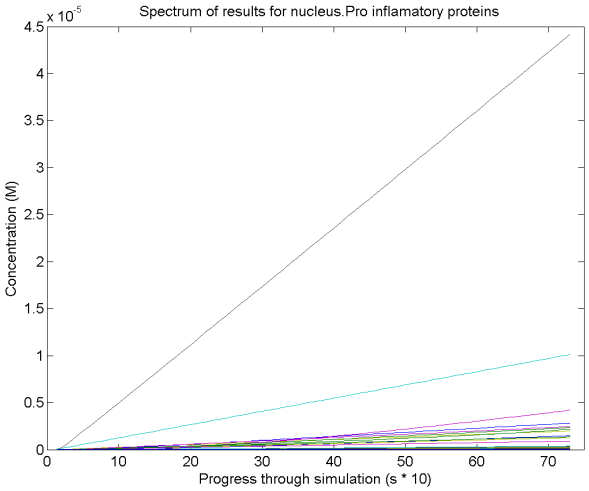

Figure 5.6: Concentration spectrum graphs for **(A)** TAK1 and **(B)** Hsp27P. We can see the Hsp27P

On the following page we include some statistics comparing integrated and non-integrated parameter sets, as well as the mean and standard deviation/mean values for the remaining parameter sets exclusive to the GC signalling pathway.

As mentioned, statistical analysis is not offered, however, the values are consistent throughout, suggesting a stable, functional model where neither of the p38 MAPK nor GC signalling pathways are dominant regarding the flux of concentration. Based on these results, our initial observations are that this integrated pathway represents a new, stable system for exploring the crosstalk between p38 MAPK and the glucocorticosteroid signalling pathway. A significant amount of model checking needs to be done to determine if the model behaves in a manner similar to that of the actual pathways, however, this is the case for all system biology models, and we hope that this further investigation can fuel research and development.

Parameter name	p38 mean	Integrated pathway mean	Ratio
main.P.kr_p38PMK2P	1.71	1.86	0.9
main.P.kr_LPS_TLR4	1.35	1.02	1.3
main.P.kr_p38_MKK6	1.53	0.94	1.6
main.P.kr_p38P_Ppase	1.39	1.20	1.2
main.P.ki_p38PMK2P	0.81	0.65	1.2
main.P.ki_MK2	1.30	0.94	1.4
main.P.kr_complex_p38P	1.14	1.30	0.9
main.P.kr_Hsp27P_Ppase	1.20	0.97	1.2
main.P.ki_p38	1.49	1.27	1.2
main.P.kc_p38ppase	12.65	16.80	0.8
main.P.kf_MK2_p3	2.27×10^{6}	1.68×10^{6}	1.4
main.P.kf_complex_p38P	1.63×10^{6}	1.85×10^{6}	0.9
main.P.kr_MKK3_complex	1.50	1.32	1.1
main.P.kc_MK2	25.86	16.38	1.6
main.P.kf_Hsp27P_Ppase	2.13×10^{6}	3.06×10^{6}	0.7
main.P.kf_MKK6_complex	1.66×10^{6}	1.48×10^{6}	1.1
main.P.kr_MKK6_complex	1.84	1.39	1.3
main.P.kc_Hsp27ppase	12.33	11.37	1.1
main.P.k_inactivation	0.69	0.79	0.9
main.P.kr_Hsp27_MK2	1.68	1.35	1.2
main.P.kf_p38_MKK6	2.00×10^{6}	2.11×10^{6}	0.9
main.P.kf_MK2P_Ppase	3.18×10^{6}	3.03×10^{6}	1.1
main.P.kc_MKK6ppas	15.91	18.13	0.9
main.P.kf_p38_MKK3	2.67×10^{6}	2.32×10^{6}	1.2
main.P.kr_MKK6P_Ppase	1.43	0.95	1.5
main.P.kc_MKK3ppase	12.88	15.58	0.8
main.P.ko_p38P	1.47	1.32	1.1
main.P.kf_MKK3P_Ppase	1.82×10^{6}	1.70×10^{6}	1.1
main.P.kc_MKK3	14.37	12.68	1.1
main.P.kr_MKK3P_Ppase	1.29	1.30	1.0
main.P.kc_p38Pcomplex	17.40	16.68	1.0
main.P.kc_MKK6	16.18	14.33	1.1
main.P.kf_p38P_Ppase	1.21×10^{6}	1.49×10^{6}	0.8
main.P.kc_complexMKK3	13.87	12.19	1.1
main.P.kf_MKK3_complex	1.63×10^{6}	2.01×10^{6}	0.8
main.P.kc_MK2ppase	19.79	19.08	1.0
main.P.kf_Hsp27_MK	1.82×10^{6}	1.54×10^{6}	1.2
main.P.kf_MKK6P_Ppase	1.87×10^{6}	1.94×10^{6}	1.0
main.P.ko_p38PMK2P	2.12	2.02	1.0
main.P.k_deg_complex	0.98	1.14	0.9
main.P.kr_MK2P_Ppase	0.96	1.10	0.9
main.P.k_activation	5.49	8.04	0.7
main.P.kf_p38P_Ppase nucleus	2.23×10^{6}	2.15×10^{6}	1.0
main.P.kc_p38	17.86	12.58	1.4
main.P.kc_complexMKK6	15.95	18.12	0.9
main.P.ko_p38	1.36	1.27	1.1
main.P.kr_MK2_p38	1.12	1.28	0.9
main.P.kr_p38_MKK3	1.13	1.18	1.0
main.P.k_reactivation	1.30	1.28	1.0
main.P.kf_LPS_TLR4	8.46E+16	5.38E+16	1.6
main.P.ko_MK2	1.10	1.54	0.7
main.P.kc_p38P_Ppase nucleus	14.80	9.83	1.5
main.P.kr_p38P_Ppas nucleus	1.00	0.87	1.2
main.P.ki_p38P	1.33	1.20	1.1

Table 5.3: p38 MAPK at 10 ng/ml LPS mean values compare to integrated pathway with 10 ng/ml LPS mean, including the p38/integrated ratio to compare them. They are consistently similar throughout, with a number of minor exceptions, although these may simply represent system variation brought about by the range of possible parameters and initial concentration

Parameter name	p38 stdev/mean	Integrated pathway stdev/mean
main.P.kr_p38PMK2P	1.73	1.57
main.P.kr_LPS_TLR4	1.99	2.18
main.P.kr_p38_MKK6	1.63	2.33
main.P.kr_p38P_Ppase	1.95	1.90
main.P.ki_p38PMK2P	2.43	2.55
main.P.ki_MK2	2.01	2.20
main.P.kr_complex_p38P	2.05	1.97
main.P.kr_Hsp27P_Ppase	2.17	2.26
main.P.ki_p38	1.74	1.86
main.P.kc_p38ppase	1.93	1.70
main.P.kf_MK2_p3	1.34	1.49
main.P.kf_complex_p38P	1.63	1.57
main.P.kr_MKK3_complex	1.75	1.94
main.P.kc_MK2	1.24	1.51
main.P.kf_Hsp27P_Ppase	1.38	1.08
main.P.kf_MKK6_complex	1.56	1.64
main.P.kr_MKK6_complex	1.66	1.87
main.P.kc_Hsp27ppase	1.88	1.98
main.P.k_inactivation	2.63	2.56
main.P.kr_Hsp27_MK2	1.79	1.86
main.P.kf_p38_MKK6	1.46	1.40
main.P.kf_MK2P_Ppase	1.05	1.05
main.P.kc_MKK6ppas	1.72	1.65
main.P.kf_p38_MKK3	1.26	1.29
main.P.kr_MKK6P_Ppase	1.77	2.16
main.P.kc_MKK3ppase	1.81	1.75
main.P.ko_p38P	1.82	1.99
main.P.kf_MKK3P_Ppase	1.65	1.60
main.P.kc_MKK3	1.93	1.85
main.P.kr_MKK3P_Ppase	1.91	1.95
main.P.kc_p38Pcomplex	1.69	1.67
main.P.kc_MKK6	1.65	1.72
main.P.kf_p38P_Ppase	1.83	1.63
main.P.kc_complexMKK3	1.87	1.93
main.P.kf_MKK3_complex	1.47	1.42
main.P.kc_MK2ppase	1.57	1.56
main.P.kf_Hsp27_MK	1.49	1.66
main.P.kf_MKK6P_Ppase	1.41	1.44
main.P.ko_p38PMK2P	1.41	1.46
main.P.k_deg_complex	2.30	2.11
main.P.kr_MK2P_Ppase	2.43	2.03
main.P.k_activation	2.81	2.69
main.P.kf_p38P_Ppase nucleus	1.35	1.36
main.P.kc_p38	1.73	1.95
main.P.kc_complexMKK6	1.65	1.52
main.P.ko_p38	1.91	1.97
main.P.kr_MK2_p38	2.18	2.06
main.P.kr_p38_MKK3	2.16	1.92
main.P.k_reactivation	2.03	2.00
main.P.kf_LPS_TLR4	2.52	3.15
main.P.ko_MK2	2.06	1.73
main.P.kc_p38P_Ppase nucleus	1.77	2.12
main.P.kr_p38P_Ppas nucleus	2.28	2.38
main.P.ki_p38P	2.06	2.07

Table 5.4: Comparing the spread around the mean for 10 ng/ml LPS p38 MAPK pathway and integrated pathway. Shows a highly similar set of values, indicating the range around the mean is comparable. This indicates that the integrated pathway is not destabilizing any of the original p38 MAPK parameters significantly, adding weight to the idea that it is a stable model.

Parameter name	MEAN	STDEV	STDEV/Mean
GR_inactive:GC association rate	2.11×10^{6}	2.87×10^{6}	1.36
GR_inactive:GC dissociation rate	1.15	2.41	2.09
GR:p38P association rate	1.71×10^{6}	2.69×10^{6}	1.57
GR:p38P dissociation rate	1.17	2.45	2.10
p38P catalytic rate on GR	13.17	25.70	1.95
GR dimerisation rate	1.9×10^{6}	2.80×10^{6}	1.48
GR un-dimerisation rate	1.35	2.74	2.03
JNK activation rate	1.29	2.48	1.92
JNK inactivation rate	1.44	2.70	1.87
JNK:AP1 association rate	1.86×10^{6}	2.75×10^{6}	1.48
JNK:AP1 dissociation rate	1.01	2.20	2.17
JNK catalytic rate on AP1	12.92	25.25	1.96
AP1P:GR association rate	2.08×10^{6}	3.02×10^{6}	1.45
AP1P:GR dissociation rate	1.28	2.41	1.88
NfkB:IkB association rate	1.77×10^{6}	2.53×10^{6}	1.43
NfkB:IkB dissociation rate	0.72	1.72	2.39
TAK1:NFkB_IkB association rate	2.24×10^{6}	3.08×10^{6}	1.37
TAK1:NFkB_IkB dissociation rate	1.22	2.46	2.01
TAK1 catalytic rate on NfkB_IkB	11.57	24.16	2.09
Spontaneous GRP dephosphorylation	1.08	2.22	2.05
Spontaneous AP1P dephosphorylation	1.22	2.57	2.11
NfkB:GR association rate	2.03×10^{6}	2.90×10^{6}	1.43
NfkB:GR dissociation rate	1.41	2.46	1.75
GR dimer nuclear import rate	0.99	2.10	2.11
GR dimer nuclear export rate	0.84	2.09	2.48
NfkB nuclear import rate	0.97	2.45	2.52
NfkB nuclear export rate	1.22	2.32	1.91
AP1P nuclear import rate	1.41	2.47	1.75
AP1P nuclear export rate	1.24	2.43	1.96
GR dimer:antiInf DNA association rate	2.27×10^{10}	3.04×10^{10}	1.34
GR dimer:antiInf DNA dissociation rate	0.01	0.02	1.93
MKP-1 protein expression rate	0.20	0.31	1.58
IkB protein expression rate	0.15	0.28	1.81
Nf-kB:proInf DNA association rate	1.80×10^{10}	2.79×10^{10}	1.55
Nf-kB:proInf DNA dissociation rate	0.01	0.03	1.95
AP1P:proInf DNA association rate	2.55×10^{10}	3.33×10^{10}	1.31
AP1P:proInf DNA dissociation rate	0.02	0.03	1.69
Pro Inf. DNA:GR dimer association	1.96×10^{10}	2.64×10^{10}	1.35
Pro Inf. DNA:GR dimer dissociation	0.01	0.02	2.48
Pro inf. protein expression1 rate	0.25	0.33	1.28
Pro inf. protein expression2 rate	0.15	0.27	1.80
IkB nuclear import rate	0.87	2.04	2.34
IkB nuclear export rate	1.66	2.99	1.80

Table 5.5: Final parameter statistics for those constants not additionally in the p38 MAPK pathway. Means and standard deviation/mean are generally consistent with previous values

Chapter 6

Conclusion and Further Work

In this final chapter, we overview the project's progress and achievements, both in a quantitative and qualitative sense. We then go on to outline the future work regarding the various elements of this project, considering both possible avenues of interests and ones being actively explored.

6.1 Project Summary

The initial motivation for this project was to develop a platform from which research into potential crosstalk between the glucocorticosteroid (GC) signalling pathway and p38 MAPK could proceed in the context of corticosteroid resistant (CSR) asthma. CSR asthma represents a significant portion of the cost and mortality associated with the disease, but affects only a tiny fraction of sufferers. By better understanding the molecular mechanism(s) and pathways causing this disease, not only may more effective treatments present themselves, but a better understanding of asthma as a whole may be obtained.

We began with a thorough overview of both possible simulation techniques and the signalling pathways associated with asthma. Included in this early stage work is a three and a half thousand word review of the state of the art regarding BIO-PEPA, included in the electronic submission of this project, although not as part of this dissertation. Based on this research, it was decided that, based on current software functionality and availability, BIO-PEPA did not represent an ideal approach to modelling complex signalling pathways. However, being able to fully justify this position was very useful, especially in the context of Barkouhk's dissertation [8].

With a good overview of the signalling processes involved, we began working with an existing signalling model developed by Hendriks et al. [30] of the p38 MAPK pathway. This pathway is involved in the pro inflammatory response, and a number of it's components are significantly up-regulated in CSR patients. While this model had already been developed, we had to reconstruct it into a format usable by MATLAB before simulations to confirm it's stability and reliability could proceed. While this was being carried out, we identified the GC signalling pathway as an ideal system to combine with the p38 MAPK pathway to investigate possible crosstalk. GCs are used as the typical treatment for asthma, although CSR patients frequently display insensitivity and up regulated p38 MAPK components. By looking at how these two pathways interact we may be able to identify putative drug targets at non-traditional locations in the pathway(s).

After an exhaustive literature review, we developed a GC signalling pathway model comparable in design, structure, size, and parameter ranges to the existing p38 MAPK pathway. During this process, it became clear that while developing the GC signalling pathway model as an independent system was necessary, integrating the two models together by hand was non-trivial. With this in mind, we identified a significant lack of existing software to integrate two SBML models together, and so set about the process of developing a complete, generic software tool to achieve just that. After identifying specification goals we used a high velocity scrum style methodology

to develop what ended up being a significant product to integrate two basic SBML[1] models together.

With both our p38 MAPK and GC models fully developed, we designed and implemented a set of MATLAB scripts to carry out a Monte Carlo based parameter generation, simulation and evaluation process on these models. Using pre-existing experimental data as the benchmark, we randomly generate a parameter set, ran the simulation using that parameter set and then evaluate how comparable the simulation outcome was with the previously reported experimental data. Initially the p38 MAPK model was used (which we had previously tested using the best fitting idealized parameter set presented by Hendriks et al.) to generate 388 parameter sets with a coefficient comparison between normalized experimental/simulation generated data of 0.88 (the best fitting Hendriks parameter set was around 0.6 when normalized). When the project began, it was envisaged that by working in parallel with Prof. Ian Adcock we would obtain empirical data relating to experimental conditions relevant to the integrated pathway, which we could then use to generate an evaluation function to identify operational parameter sets for the integrated model in the same way as we had shown for the p38 MAPK model. Unfortunately this data was not available, however, we were able to generate some entirely synthetic data and use the p38 MAPK experimental data to evaluate that our integrated model was functional and produced plausible output, even if the actual data was not biologically relevant. With this successfully done, the work required to generate correctly evaluated parameters for the integrated pathway is simply a case of entering in the new empirical data running the scripts.

6.2 Future Work

As mentioned in the preceding section, this project comprises of related yet independent components;

- Reconstruction of the p38 MAPK pathway model, originally developed by Hendriks et al [30] into a more efficient, syntactically correct MATLAB compliant SBML model
- A totally novel SBML implementation of the GC signalling pathway based on a significant amount of data and good understanding of the underlying biochemical processes
- A software tool for integrating two SBML models together in a semantically and syntactically correct manner
- A set of MATLAB scripts for automating the process of generating random parameters, running a simulation with those parameters, assessing if the generated model meets the experimental criterion and then either discarding or saving the parameter sets

We consider each of these four elements in turn, evaluating future endeavours, as well as finally considering the extension of this project.

6.2.1 p38 MAPK Model

While semantically this model was already in the public domain, it was in a not easily usable format. Despite being described as SBML, the original file failed to comply to either SBML or MATLAB specifications for model manipulation. We do not presume to suggest that without this work further development and work based on this model would not have been possible, however, it significantly reduces the friction associated with justifying future projects using the model. The p38 MAPK pathway is central to a significant number of biological process, and is especially relevant in the prognosis and diagnosis of cancer[17]. With our SBMLIntegrator software, having an operational and widely available version of the MAPK model may facilitate future investigation

[1]Basic here denotes a model which contains one or more of the SBML elements: Unit Definition, Compartment, Species, Parameter, Rule or Reaction

into cancer-related signalling. We plan to discuss the possible submission of this updated version to the BioModels database with Hendriks et al.

6.2.2 Glucocorticosteriod Signalling Pathway

To our knowledge there are no GC signalling models which convey the detail of ours. Additionally, our approach of modelling gene expression has not been seen in any other SBML models either. Clearly a great deal of additional work needs to be done to verify that this model and the integration model are valid, but using SBMLIntegrator we can easily make changes to the seperate GC signalling pathway and re-integrate the two. It is hoped that when the empirical data produced by Prof. Adcock becomes available we will be able to begin a significant and more advanced parameter estimation process on the integrated pathway. Based on the results if these simulations, the presence of pathway crosstalk should be easily identifiable. It is important to remember that crosstalk in this model does not necessarily mean that biological models participate in crosstalk. However, it may indicate potential future experimental targets, which could confirm or deny it's existence.

6.2.3 SBMLIntegrator

The SBMLIntegrator software is currently available through github [2] and is in it's alpha stage of public release. Over the next few weeks we will add in additional functionality relating to the SBML elements not already covered by the implementation (function definitions, constraints, initial assignments and events). In addition, we are in the process of discussing with the SBML community features for a good, fast GUI based model designer and integrator[3]. By gauging specific desires of those who use the software most, we will attempt to use the underlying SBMLIntegrator core to construct an SBML modelling tool. There is discussion with other developers around the world regarding pooling resources, and we hope to have a functional version of this software up and running my mid October.

6.2.4 Parameter Generation and Evaluation

Our MATLAB tools for parameter estimation have provided an efficient, fast and effective way to generate a widely ranging set of parameters for our model estimation. However, there are issues associated with the number distribution of the random values generated, but with the ranges provided a uniformly distributed random number creates such a vast search space that a simple, Monte-Carlo approach is not tractable for a system of this size. We would suggest, therefore, that the development of intelligent parameter estimation algorithms represents a significant challenge facing not only this project, but the entire parameter estimation space. Using a guided Monte Carlo approach may facilitate favourable results, whereby a search space is narrowed based on a dynamic programming approach in a top down manner. Such an algorithm would have a significant impact on the systems biology discipline as a whole, and could allow for automated parameter estimation for models on the BioModels database.

6.3 Final Project Work

As a final note, work has begun on a paper detailing the results of this project. We focus on the development of the glucocorticosteriod model and it's integration with the p38 MAPK model. If the empirical data becomes available during this write up process it may be possible to incorporate some of the results and analysis here. However, irrespectively, it will provide an example of the development of an associated signalling pathway based on an existing one, and the progress of developing a system where the pathway interface can be explored. We hope that by providing the tools to carry out this integration that there will be an increase in the research into signalling crosstalk, an area where systems biology has the potential to yield great results.

[2] https://github.com/rednaxela/SBMLIntegrator

[3] http://sbml.org/Forums/index.php?t=msg&goto=7160&rid=6678#msg_7160

Appendix A

SBMLIntegrator

A.1 SBMLIntegrator Output Screens

A.1.1 Explore Model

```
--------------------------------------
NB: Model
Select components to show
--------------------------------------
[1] ------------ Function Definitions
[2] ------------ Unit Definitions
[3] ------------ Compartments
[4] ------------ Species
[5] ------------ Parameters
[6] ------------ Rules
[7] ------------ Initial Assignments
[8] ------------ Constraints
[9] ------------ Reactions
[10] ----------- Events
[11] ----------- Return
Select:
```

Interface 11: Explore model initial screen - from here each option goves a list of the various elements and their attributes. If a value is abscent from an attribute we display the fact with message

A.1.2 Display Summary

```
##############################################################
####################   Display Summary   #####################
##############################################################
File name ---------------- p38model.xml
Model ID ----------------
Model name ---------------
Model version ------------ 1
ModeL level -------------- 2
Model substance units ----
Model time units ---------
Model volume units -------
Model area units ---------
Model length units -------
Model extent units -------
Model conversion factor --
---------------------------------
Summary of model components
---------------------------------
No of funtions ----------- 0
No of unit definitions --- 19
No of compartments ------- 4
No of species ------------ 40
No of parameters --------- 169
No of initial assigments - 0
No of rules -------------- 101
No of constraints -------- 0
No of reactions ---------- 35
No of events ------------- 0
---------------------------------
```

Interface 12: Model summary screen, with the original p38-MAPK model as the example

A.1.3 Display Compartments

```
##############################################################
##################   Display Compartments   ##################
##############################################################
There are 4 compartments defined;
Compartment [1] ID(main_cytosol) name(cytosol)
    Spatial Dimensions: 3
    Size: 1
    Units: litre
    Constant: 1
Compartment [2] ID(main_Medium) name(Medium)
    Spatial Dimensions: 3
    Size: 1
    Units: litre
    Constant: 1
```

Interface 13: Compartment viewer, describing the two of the compartments in the original p38 MAPK model

A.1.4 Display Reactions

```
##############################################################
####################   Display Reaction   ####################
##############################################################
There are 35 reactions defined;
Reaction [1] Name(Rxn1271) ID(main_cytosol_Rxn1271)
    main_cytosol_complex_MKK6 <---> main_cytosol_complex + main_cytosol_MKK6P

    Modfiers: <none>
    Rate law: main_cytosol_Rxn1271_Flux
      Local parameters: 0
      Global parameters: 0
      Referenced rule: main_cytosol_Rxn1271_Kf * main_cytosol_complex_MKK6
      * main_cytosol
```

Interface 14: Reaction viewer, describing the one of the reactions in the original p38 MAPK model

A.1.5 Display Rules

```
################################################################
#####################   Display Rules   ########################
################################################################
There are 101 rules defined;
Rule [1] ID(main_cytosol_Rxn1534_Kf)
    Assignment rule: main_P_kf_LPS_TLR4 * main_cytosol
    Units: litre^(2) * second^(-1) * mole^(-1)

Rule [2] ID(main_cytosol_Rxn126_Kr)
    Assignment rule: main_P_kr_p38P_Ppase
    Units: second^(-1)

Rule [3] ID(main_nucleus_Rxn1_Kr)
    Assignment rule: main_P_kr_MK2_p38
    Units: second^(-1)
```

Interface 15: Rules viewer, describing three of the rules in the original p38 MAPK model

Appendix B

Model development

B.1 p38 MAPK Initial Concentration Ranges

Compartment	Name	Lower conc	Upper conc
Medium	LPS	0.00001	0.01
Cytoplasm	TLR4	0.25	2.25
Cytoplasm	MKK3	0.01	0.09
Cytoplasm	MKK6	0.01	0.12
Cytoplasm	p38	1.3	11.7
Cytoplasm	Hsp27	6.5	58.8
Cytoplasm	PhosphataseMKK3	10-3	102
Cytoplasm	PhosphataseMKK6	10-3	102
Cytoplasm	Phosphatasep38	10-3	102
Cytoplasm	PhosphataseMK2	10-3	102
Cytoplasm	PhosphataseHsp27	10-3	102
Nucleus	Phosphatasep38,nucleus	10-3	102
Nucleus	MK2nucleus	21	190
Cytoplasm	LPS:TLR4	0	0
Cytoplasm	TAK1complex	0	0
Cytoplasm	p38P:TAK1complex	0	0
Cytoplasm	TAK1complexinactive	0	0
Cytoplasm	TAK1complex:MKK3	0	0
Cytoplasm	MKK3P	0	0
Cytoplasm	Ppase:MKK3P	0	0
Cytoplasm	TAK1complex:MKK6	0	0
Cytoplasm	MKK6P	0	0
Cytoplasm	Ppase:MKK6P	0	0
Cytoplasm	MKK3P:p38	0	0
Cytoplasm	MKK6P:p38	0	0
Cytoplasm	p38P	0	0
Cytoplasm	p38P:MK2P	0	0
Cytoplasm	MK2P	0	0
Cytoplasm	Ppase:MK2P	0	0
Cytoplasm	MK2P:Hsp27	0	0
Cytoplasm	Hsp27P	0	0
Cytoplasm	Ppase:Hsp27	0	0
Nucleus	Ppase:p38Pnucleus	0	0
Nucleus	p38Pnucleus	0	0
Nucleus	p38P:MK2nucleus	0	0
Nucleus	p38P:MK2Pnucleus	0	0

Table B.1: Initial concentration ranges for p38 model (and by association p38 model components in the p38-GC model). Based on data from Hendriks et al[30]. Units in μM

B.2 p38 MAPK Parameter Ranges

Name	Description	Rxn	Lower	Upper	Units
$Volume_{cytosol}$	Cytosolic volume		10^{-12}	10^{-12}	L
$volumenucleus$	Nuclear volume		10^{-13}	10^{-13}	L
$k_{f,LPS:TLR4}$	LPS:TLR4 association rate	1	10^{7}	10^{18}	M^{-1} min^{-1}
$k_{r,LPS:TLR4}$	LPS:TLR4 dissociation rate	1	10^{-4}	10^{1}	min^{-1}
$k_{activation}$	LPS:TLR4 -> TAK1complex activation rate	2	10^{-4}	10^{2}	min^{-1}
$k_{deactivation}$	TAK1complex deactivation rate	2	10^{-4}	10^{1}	min^{-1}
$k_{reactivation}$	$TAK1complex_{inactive}$ reactivation rate	4	10^{-4}	10^{1}	min^{-1}
$k_{deg,complex}$	TAK1complex degradation rate	3	10^{-4}	10^{1}	min^{-1}
$k_{f,complex:p38P}$	TAK1complex:phospho-p38 association rate	5	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,complex:p38P}$	TAK1complex:phospho-p38 dissociation rate	5	10^{-4}	10^{1}	min^{-1}
$k_{c,p38P:complex}$	phospho-p38 catalytic rate for TAK1complex	6	10^{-2}	10^{2}	min^{-1}
$k_{f,MKK3:complex}$	TAK1complex:MKK3 association rate	11	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MKK3:complex}$	TAK1complex:MKK3 dissociation rate	11	10^{-4}	10^{1}	min^{-1}
$k_{c,complex(MKK3)}$	TAK1complex catalytic rate for MKK3	12	10^{-2}	10^{2}	min^{-1}
$k_{f,MKK3P:Ppase}$	phospho-MKK3:$Phosphatase_{MKK3}$ association rate	14	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MKK3:Ppase}$	phospho-MKK3:$Phosphatase_{MKK3}$ dissociation rate	14	10^{-4}	10^{1}	min^{-1}
$k_{c,PpaseMKK3}$	$Phosphatase_{MKK3}$ catalytic rate	13	10^{-2}	10^{2}	min^{-1}
$k_{f,MKK6:complex}$	TAK1complex:MKK6 association rate	7	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MKK6;complex}$	TAK1complex:MKK6 dissociation rate	7	10^{-4}	10^{1}	min^{-1}
$k_{c,complex(MKK6)}$	TAK1complex catalytic rate for MKK6	8	10^{-2}	10^{2}	min^{-1}
$k_{f,MKK6P:Ppase}$	phospho-MKK6:$Phosphatase_{MKK6}$ association rate	10	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MKK6P:Ppase}$	phospho-MKK6:$Phosphatase_{MKK6}$ dissociation rate	10	10^{-4}	10^{1}	min^{-1}
$k_{c,PpaseMKK6}$	$Phosphatase_{MKK6}$ catalytic rate	9	10^{-2}	10^{2}	min^{-1}
$k_{f,p38:MKK3}$	phospho-MKK3:p38 association rate	15	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,p38:MKK3}$	phospho-MKK3:p38 dissociation rate	15	10^{-4}	10^{1}	min^{-1}
$k_{c,MKK3}$	phospho-MKK3 catalytic rate	16	10^{-2}	10^{2}	min^{-1}
$k_{f,p38:MKK6}$	phospho-MKK6:p38 association rate	17	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,p38:MKK6}$	phospho-MKK6:p38 dissociation rate	17	10^{-4}	10^{1}	min^{-1}
$k_{c,MKK6}$	phospho-MKK6 catalytic rate	18	10^{-2}	10^{2}	min^{-1}
$k_{f,p38P:Ppase}$	phospho-p38:$Phosphatase_{p38}$ association rate (cytosol)	20	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,p38P:Ppase}$	phospho-p38:$Phosphatasep38_{p38}$ dissociation rate (cytosol)	20	10^{-4}	10^{1}	min^{-1}
$k_{c,Ppasep38}$	$Phosphatasep_{p38}$ catalytic rate (cytosol)	19	10^{-2}	10^{2}	min^{-1}
$k_{f,p38P:Ppase,nucleus}$	phospho-p38:Phosphatasep38 association rate (nucleus)	21	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,p38P:Ppase,nucleus}$	phospho-p38:Phosphatasep38 dissociation rate (nucleus)	21	10^{-4}	10^{1}	min^{-1}
$k_{c,p38P:Ppase,nucleus}$	Phosphatasep38 catalytic rate (nucleus)	22	10^{-2}	10^{2}	min^{-1}
$k_{f,MK2:p38}$	phospho-p38:MK2 association rate	23	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MK2:p38}$	phospho-p38:MK2 dissociation rate	23	10^{-4}	10^{1}	min^{-1}
$k_{c,p38}$	phospho-p38 catalytic rate for MK2	24	10^{-2}	10^{2}	min^{-1}
$k_{r,p38P:MK2P}$	phospho-p38:phospho-MK2 dissociation rate	25	10^{-4}	10^{1}	min^{-1}
$k_{f,MK2P:Ppase}$	phospho-MK2:$Phosphatase_{MK2}$ association rate	26	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,MK2P:Ppase}$	phospho-MK2:$Phosphatase_{MK2}$ dissociation rate	26	10^{-4}	10^{1}	min^{-1}
$k_{c,PpaseMK2}$	$Phosphatase_{MK2}$ catalytic rate	27	10^{-2}	10^{2}	min^{-1}
$k_{f,Hsp27:MK2}$	phospho-MK2:Hsp27 association rate	28	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,Hsp27:MK2}$	phospho-MK2:Hsp27 dissociation rate	28	10^{-4}	10^{1}	min^{-1}
$k_{c,MK2}$	phospho-MK2 catalytic rate	29	10^{-2}	10^{2}	min^{-1}
$k_{f,Hsp27P:Ppase}$	phospho-Hsp27:$Phosphatase_{Hsp27}$ association rate	30	10^{4}	10^{7}	M^{-1} min^{-1}
$k_{r,Hsp27P:Ppase}$	phospho-Hsp27:$Phosphatase_{Hsp27}$ dissociation rate	30	10^{-4}	10^{1}	min^{-1}
$k_{c,PpaseHsp27}$	$Phosphatase_{Hsp27}$ catalytic rate	31	10^{-2}	10^{2}	min^{-1}
$k_{i,p38}$	p38 nuclear import rate	32	10^{-4}	10^{1}	min^{-1}
$k_{o,p38}$	p38 nuclear export rate	32	10^{-4}	10^{1}	min^{-1}
$k_{i,p38P}$	phospho-p38 nuclear import rate	33	10^{-4}	10^{1}	min^{-1}
$k_{o,p38P}$	phospho-p38 nuclear export rate	33	10^{-4}	10^{1}	min^{-1}
$k_{i,p38P:MK2P}$	phospho-p38:phospho-MK2 nuclear co-import rate	34	10^{-4}	10^{1}	min^{-1}
$k_{o,p38P:MK2P}$	phospho-p38:phospho-MK2 nuclear co-export rate	34	10^{-4}	10^{1}	min^{-1}
$k_{i,MK2}$	MK2 nuclear import rate	35	10^{-4}	10^{1}	min^{-1}
$k_{o,MK2}$	MK2 nuclear export rate	35	10^{-4}	10^{1}	min^{-1}

Table B.2: p38 MAPK parameter ranges

Appendix C

Simulation results

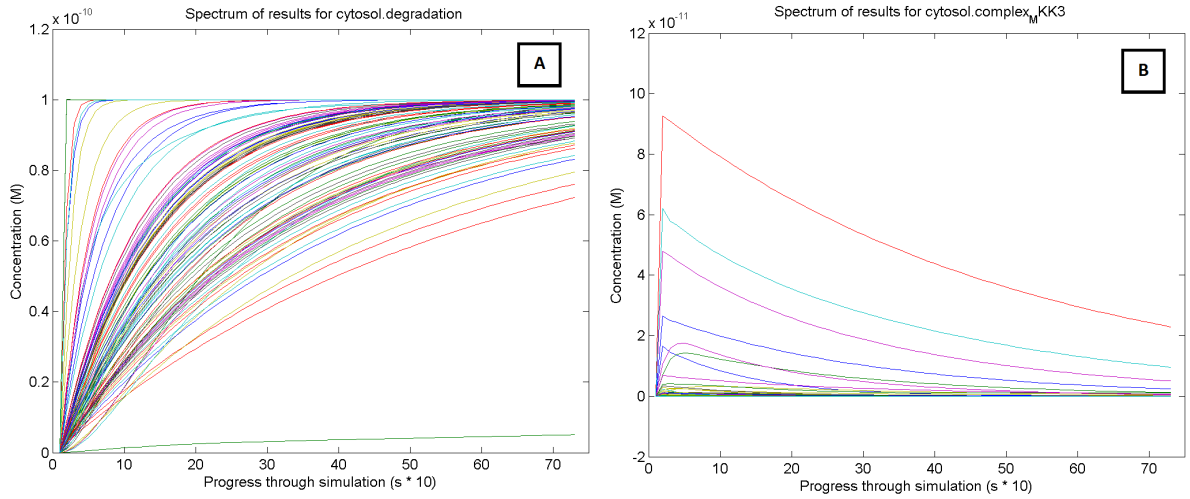

Figure C.1: **(A)** - Cytosol degradation concentration spectrum graph and **(B)** - TAK1:MKK3 concentration spectrum graphs, both at 10ng/ml

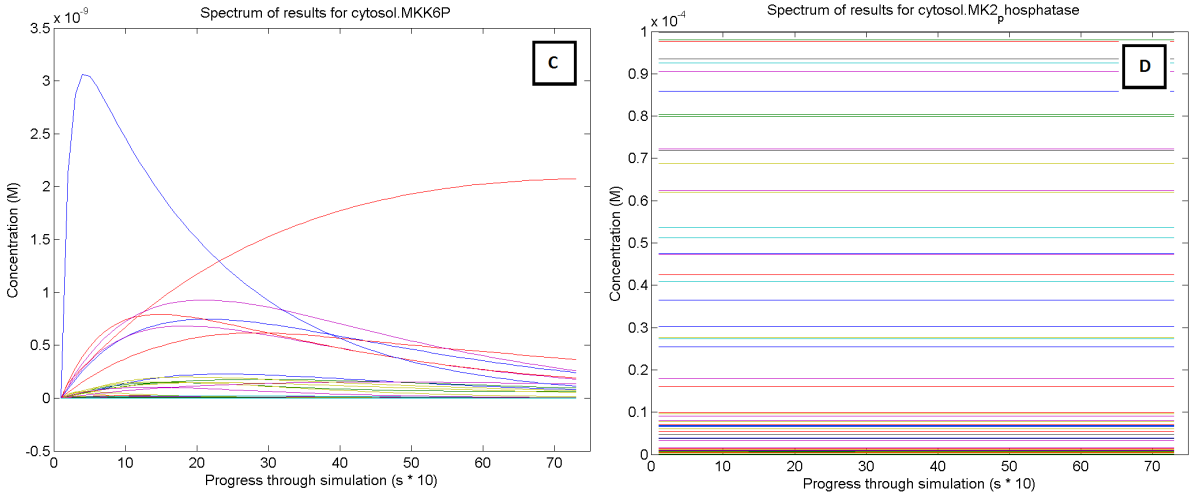

Figure C.2: **(C)** - MKK6 concentration spectrum graph and **(D)** - MK2_phosphatase concentration spectrum graphs, both at 10ng/ml

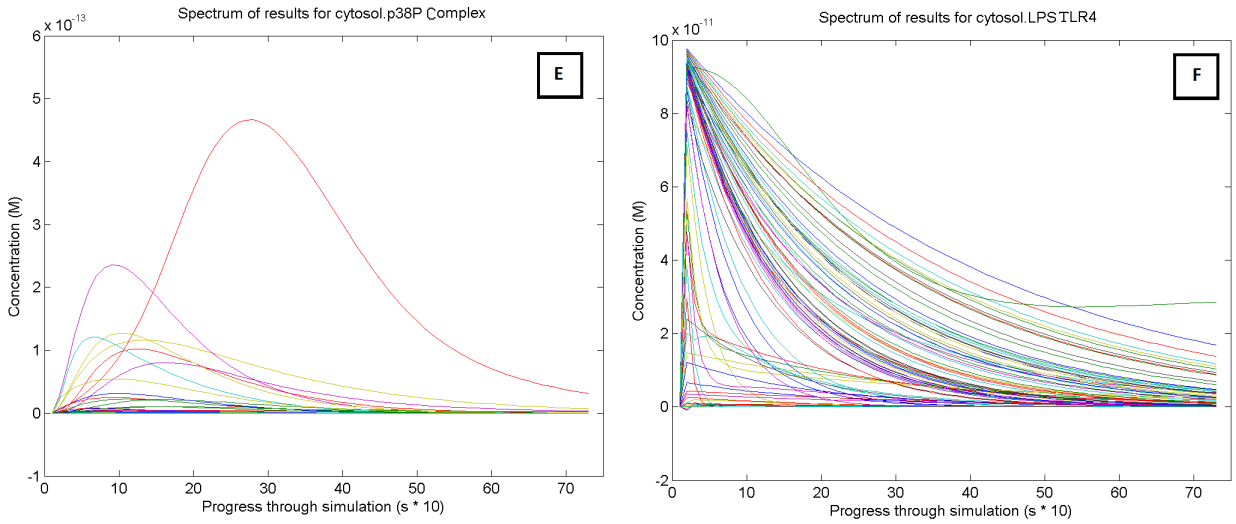

Figure C.3: **(E)** - p38P:TAK1 concentration spectrum graph and **(F)** - LPS:TLR4 concentration spectrum graphs, both at 10ng/ml

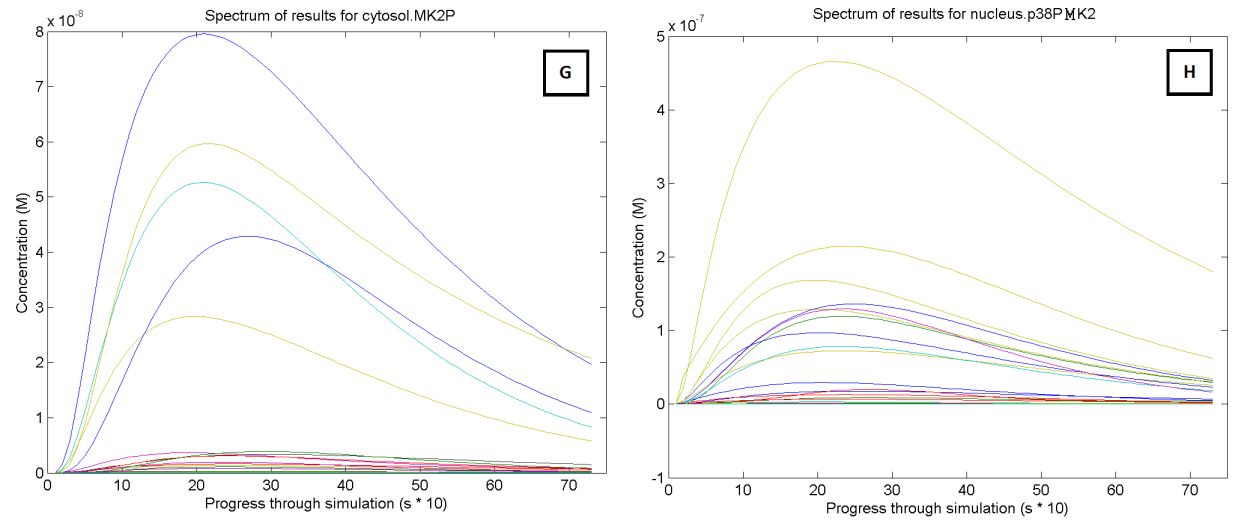

Figure C.4: **(G)** - MK2P concentration spectrum graph and **(H)** - Nuclear localized p38P:MK2concentration spectrum graphs, both at 10ng/ml

Acronyms

AP-1 Activator Protein-1. 20

CDK Cylcin Dependent Kinase. 21

CSR Corticosteroid Resistant. 4, 21, 22, 79

CTMC Continous Time Markov Chain. 11–13

DNA Deoxyribonucleic Acid. 6

ERK Extracellular Regulated Kinase. 20

GC Glucocorticosteroid. 4, 5, 19, 22, 25, 51, 59, 60, 62, 63, 68, 74, 75, 79–81

GR Glucocortocoid Receptors. 20, 22, 54, 62

GRE Glucocortocoid Response Element. 20

GSK-3 Glycogen Synthase Kinase 3. 21

hsp Heat Shock Protein. 20

IL Interleukin. 19

IL-13 Interleukin-13. 51

IL-2 Interleukin-2. 51

IL-4 Interleukin-4. 51

IRI Import Replace Integrate. 29, 41

JNK c-Jun N-terminal Kinase. 20, 21

LABAs Long Acting β Agonists. 19

MAPK Mitogen Activated Protein Kinase. 20, 21

MCA Metabolic Control Analysis. 6

MKP-1 MAPK phosphatase-1. 20

mRNA Messenger Ribonucleic Acid. 8

MWC Monod Wyman Changeux. 11

ODEs Ordinary Differential Equations. 12

PBMCs Peripheral Blood Mononuclear Cells. 21

PPI Protein Protein Interaction. 9

RNA xRibonucleic Acid. 6

SBML Systems Biology Markup Language. 4, 16

SPA Stochastic Process Algebra. 11

TNF Tumour Necrosis Factor. 19

XML Extensible Markup Language. 16

Bibliography

[1] Devrim Acehan, Xuejun Jiang, David Gene Morgan, John E Heuser, Xiaodong Wang, and Christopher W Akey. Three-dimensional structure of the apoptosome: Implications for assembly, procaspase-9 binding, and activation. Molecular Cell, 9(2):423–432, February 2002.

[2] I M Adcock, S J Lane, C R Brown, T H Lee, and P J Barnes. Abnormal glucocorticoid receptor-activator protein 1 interaction in steroid-resistant asthma. The Journal of Experimental Medicine, 182(6):1951–1958, December 1995.

[3] Ian M. Adcock and Peter J. Barnes. Molecular mechanisms of corticosteroid resistance. CHEST, 134-2:394–401, 2008.

[4] Ian M Adcock and Gaetano Caramori. Cross-talk between pro-inflammatory transcription factors and glucocorticoids. Immunol Cell Biol, 79(4):376–384, August 2001.

[5] A.S. Baldwin Jr. Series introduction: the transcription factor nf-kappab and human disease. J Clin Invest, 107:3–6, 2001.

[6] Peter Barnes and Michael Karin. Nuclear factor-kb: a pivotal transcription factor in chronic inflammatory diseases. The New England journal of medicine, 336 : 15:1066, 1997.

[7] Peter J. Barnes. Efficacy of inhaled corticosteroids in asthma. Journal of Allergy and Clinical Immunology, 102(4):531–538, October 1998.

[8] C. Baroukh. Representing biological processes using process calculus. Master's thesis, Imperial College London, 2010.

[9] Falko Baus and Pieter S Kritzinge. Stochastic Petri Nets - An Introduction to the Theory. Bause and Kritzinger, 2002.

[10] P. Bhavsar, N. Khorasani, M. Hew, M. Johnson, and K. F. Chung. Effect of p38 mapk inhibition on corticosteroid suppression of cytokine release in severe asthma. European Respiratory Journal, 35(4):750–756, April 2010.

[11] Falk Schreiber Björn H. Junker. Analysis of Biological Networks. Wiley, 2008.

[12] Corren J Bensch G Busse WW Whitmore J Borish LC, Nelson HS. Efficacy of soluble il-4 receptor for the treatment of adults with asthma. J Allergy Clin Immunol, 107:963–970, 2001.

[13] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible markup language (xml) 1.0 (fifth edition), November 2008.

[14] Matthew D. Brown and David B. Sacks. Protein scaffolds in map kinase signalling. Cellular Signalling, 21(4):462–469, April 2009.

[15] M Brunner and H. Bujard. Promoter recognition and promoter strength in the escherichia coli system. EMBO J, 6(10):3139–3144, 1987.

[16] Federica Ciocchetta and Jane Hillston. Bio-pepa: An extension of the process algebra pepa for biochemical networks. Electron. Notes Theor. Comput. Sci, 194:3:103–117, 2008.

[17] A S Dhillon, S Hagan, O Rath, and W Kolch. Map kinase signalling pathways in cancer. Oncogene, 26(22):3279–3290, 0000.

[18] Danielle Duma, Christine M. Jewell, and John A. Cidlowski. Multiple glucocorticoid receptor isoforms and mechanisms of post-translational modification. The Journal of Steroid Biochemistry and Molecular Biology, 102(1-5):11–21, December 2006.

[19] R Duncan and EH McConkey. How many proteins are there in a typical mammalian cell? Clin Chem, 28(4):749–755, April 1982.

[20] T. Kino E. Charmandari. Novel causes of generalized glucocorticoid resistance. Horm Metab Res, 39(6):445–450, 2007.

[21] Michael R. Edwards, Nathan W. Bartlett, Deborah Clarke, Mark Birrell, Maria Belvisi, and Sebastian L. Johnston. Targeting the nf-[kappa]b pathway in asthma and chronic obstructive pulmonary disease. Pharmacology & Therapeutics, 121(1):1–13, January 2009.

[22] D. Fell. Metabolic control analysis: a survey of its theoretical and experimental development. Biochem J., 286 Part-2:313–330, 1992.

[23] D. Gilbert and M. Heiner. From petri nets to differential equations - an integrative approach for biochemical network analysis. Lecture Notes in Computer Science, 4024:181–200, 2006.

[24] D. Gillespie. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem, 81:2340–2361, 1977.

[25] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of Computational Physics, 22:403, 1976.

[26] Peter J. E. Goss and Jean Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. Proceedings of the National Academy of Sciences, 95(12):6750–6755, June 1998.

[27] Maria Luisa Guerriero. Qualitative and quantitative analysis of a bio-pepa model of the gp130/jak/stat signalling pathway. Lecture Notes in Computer Science, 5750:90–115, 2009.

[28] C.M. Guldberg and P. Waage. Uber die chemische affinitat. Journal fur Pracktische Chemie, 19:69, 1879.

[29] S Lawrence Zipursky Paul Matsudaira David Baltimore Harvey Lodish, Arnold Berk and James Darnell. Molecular Cell Biology, 4th edition. New York: W. H. Freeman, 2000. ISBN-10: 0-7167-3136-3.

[30] Bart S. Hendriks, Fei Hua, and Jeffrey R. Chabot. Analysis of mechanistic pathway models in drug discovery: p38 pathway. Biotechnology Progress, 24(1):96–109, 2008.

[31] A. V. Hill. The possible effects of the aggregation of the molecules of huemoglobin on its dissociation curves. Proceedings of the Physilogical Society (Sup), 40:4–7, 1910.

[32] C. A. R. Hoare. Communicating sequential processes. Commun. ACM, 21(8):666–677, 1978.

[33] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. J Physiol., 117(4):500–544, 1952.

[34] Michael Hucka, Frank T. Bergmann, Stefan Hoops, Sarah M. Keating, Sven Sahle, James C. Schaff, Lucian P. Smith, and Darren J. Wilkinson. The systems biology markup language (sbml): Language specification for level 3 version 1 core. Nature Precedings, http://dx.doi.org/10.1038/npre.2010.4959.1:–, 2010.

[35] Elvis Irusen, John G. Matthews, Atsushi Takahashi, Peter J. Barnes, Kian F. Chung, and Ian M. Adcock. p38 mitogen-activated protein kinase-induced glucocorticoid receptor phosphorylation reduces its activity: Role in steroid-insensitive asthma. Journal of Allergy and Clinical Immunology, 109(4):649–657, April 2002.

[36] K. Ito, KF. Chung, and IM. Adcock. Update on glucocorticoid action and resistance. J Allergy Clin Immunol, 117:522–543, 2006.

[37] Eric Jacques, Abdelhabib Semlali, Louis-Philippe Boulet, and Jamila Chakir. Ap-1 overexpression impairs corticosteroid inhibition of collagen production by fibroblasts isolated from asthmatic subjects. American Journal of Physiology - Lung Cellular and Molecular Physiology, –:–, June 2010.

[38] John L Tymoczko Jeremy M Berg and Lubert Stryer. Biochemistry, 5th edition. New York: W H Freeman, 2002.

[39] H. Kacser and J. A. Burns. The control of flux. Symp Soc Exp Biol., 27:65–104, 1973.

[40] A. D. King, N. Pržulj, and I. Jurisica. Protein complex prediction via cost-based clustering. Bioinformatics, 20(17):3013–3020, November 2004.

[41] Tomoshige Kino, Takamasa Ichijo, Niranjana D. Amin, Sashi Kesavapany, Yonghong Wang, Nancy Kim, Sandesh Rao, Audrey Player, Ya-Li Zheng, Michael J. Garabedian, Ernest Kawasaki, Harish C. Pant, and George P. Chrousos. Cyclin-dependent kinase 5 differentially regulates the transcriptional activity of the glucocorticoid receptor through phosphorylation: Clinical implications for the nervous system response to glucocorticoids and stress. Molecular Endocrinology, 21(7):1552–1568, July 2007.

[42] J.C. Kips, B.J. O'Connor, S.J. Langley, A. Woodcock, H.A. Kerstjens, and D.S Postma. Effect of sch55700, a humanized anti-human interleukin-5 antibody, in severe persistent asthma: a pilot study. Am J Respir Crit Care Med, 167:1655–1659, 2003.

[43] J. Koomey. Why we can expect ever more amazing mobile computing devices in the years ahead (presentation), 2011.

[44] C. Kuttler and J. Niehren. Gene regulation in the pi calculus: simulating cooperativity at the lambda switch. Transactions on Computational Systems Biology, 4230-VII:24–55, 2004.

[45] SJ Lane, BA Atkinson, R Swaminathan, and TH Lee. Hypothalamic-pituitary-adrenal axis in corticosteroid-resistant bronchial asthma. Am. J. Respir. Crit. Care Med., 153(2):557–560, February 1996.

[46] SJ. Lane, JB Palmer., IF. Skidmore, and TH. Lee. Corticosteroid pharmacokinetics in asthma. Lancet., 17;336(8725):1265, 1990 Nov.

[47] D Y Leung, R J Martin, S J Szefler, E R Sher, S Ying, A B Kay, and Q Hamid. Dysregulation of interleukin 4, interleukin 5, and interferon gamma gene expression in steroid-resistant asthma. The Journal of Experimental Medicine, 181(1):33–40, January 1995.

[48] Donald Y. M. Leung and John W Bloom. Update on glucocorticoid action and resistance. Journal of Allergy and Clinical Immunology, 111(1):3–22, January 2003.

[49] Mathworks. Matlab simbiology package, 2011.

[50] L. Menten and M.I. Michaelis. Die kinetik der invertinwirkung. Biochem Z, 49:333–369, 1913.

[51] Aaron L. Miller, M. Scott Webb, Alicja J. Copik, Yongxin Wang, Betty H. Johnson, Raj Kumar, and E. Brad Thompson. p38 mitogen-activated protein kinase (mapk) is a key mediator in glucocorticoid-induced apoptosis of lymphoid cells: Correlation between p38 mapk activation and site-specific phosphorylation of the human glucocorticoid receptor at serine 211. Mol Endocrinol, 19(6):1569–1583, June 2005.

[52] Kenneth Murphy. Immunobiology. Garlander Science, 2011.

[53] Brian M. Necela and John A. Cidlowski. Mechanisms of glucocorticoid receptor action in noninflammatory and inflammatory cells. Proc Am Thorac Soc, 1(3):239–246, November 2004.

[54] NHLBI. Asthma overview, Feb 2011.

[55] P.M. O'Byrne, M.D. Inman, and E. Adelroth. Reassessing the th2 cytokine basis of asthma. Trends Pharmacol Sci, 25:244–248, 2004.

[56] BioModels Database A Database of Annotated Published Models. http://www.ebi.ac.uk/biomodels-main/.

[57] KEGG: Kyoto Encyclopedia of Genes and Genomes. http://www.genome.jp/kegg/.

[58] Andrea Degasperi Ozgur E. Akman, Federica Ciocchetta and Maria Luisa Guerriero. Modelling biological clocks with bio-pepa: Stochasticity and robustness for the neurospora crassa circadian network. Lecture Notes in Computer Science, 5688:52–67, 2009.

[59] Klaus Paal, Patrick A. Baeuerle, and M. Lienhard Schmitz. Basal transcription factors tbp and tfiib and the viral coactivator e1a 13s bind with distinct affinities and kinetics to the transactivation domain of nf-kb p65. Nucleic Acids Research, 25(5):1050–1055, March 1997.

[60] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with namd. J. Comput. Chem., 26(16):1781–1802, 2005.

[61] M. Rathinam, L. Petzold, Y. Cao, and D. Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. The Journal of Chemical Physics, 119:12784, 2003.

[62] Zoltán Szatmáry, Michael J. Garabedian, and Jan Vilček. Inhibition of glucocorticoid receptor-mediated transcriptional activation by p38 mitogen-activated protein (map) kinase. Journal of Biological Chemistry, 279(42):43708–43715, October 2004.

[63] W. TRAUB, A. YONATH, and D. M. SEGAL. On the molecular structure of collagen. Nature, 221(5184):914–917, March 1969.

[64] Fyodor D. Urnov and Alan P. Wolffe. Chromatin remodeling and transcriptional activation: the cast (in order of appearance). Oncogene, 20:2991–3006, 2001.

[65] Dimitri van Heesch. Doxygen - generate documentation from source code (http://www.stack.nl/ dimitri/doxygen/).

[66] Donald Voet and Judith G. Voet. Biochemistry, 4th Edition. Wiley, 2011.

[67] D. C. Walker, J. Southgate, G. Hill, M. Holcombe, D. R. Hose, S. M. Wood, S. Mac Neil, and R. H. Smallwood. The epitheliome: agent-based modelling of the social behaviour of cells. Biosystems, 76(1-3):89–100, August.

[68] Zhen Wang, Weiwei Chen, Evelyn Kono, Thoa Dang, and Michael J. Garabedian. Modulation of glucocorticoid receptor phosphorylation and transcriptional activity by a c-terminal-associated protein phosphatase. Molecular Endocrinology, 21(3):625–634, March 2007.

[69] Nancy L. Weigel and Nicole L. Moore. Steroid receptor phosphorylation: A key modulator of multiple receptor functions. Molecular Endocrinology, 21(10):2311–2319, October 2007.

[70] Wikipedia. Petri net image, 2011.

[71] Marsha Wills-Karp, Jackie Luyimbazi, Xueying Xu, Brian Schofield, Tamlyn Y. Neben, Christopher L. Karp, and Debra D. Donaldson. Interleukin-13: Central mediator of allergic asthma. Science, 282(5397):2258–2261, December 1998.