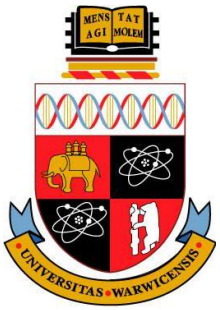

Derivation of Process Algebraic Models of Biochemical Systems

by

Ritesh V. Krishna

A thesis submitted to
The Department of Computer Science
in candidacy for the degree of
Master of Science by Research

Academic Supervisor: *Dr. Sara Kalvala*

University of Warwick, Department of Computer Science
December, 2005

Abstract

Biochemical pathways have traditionally been modeled using ordinary differential equations (ODEs), and this approach has resulted in a huge knowledge-base of inter-species interaction mechanisms found in biological systems. However, differential equation based modeling has a few disadvantages and it has been argued that a new perspective of a system can be derived by looking at its stochastic variant. On the other hand, collaboration between computer scientists and biologists has resulted in the application of process-algebras for modeling of biological systems, which allows these systems to be seen as concurrent and communicating sets of independent agents trying to achieve a common goal. Process-algebra based modeling has several advantages of its own and is gaining popularity among researchers. A problem that is apparent is that many of the biological models that exist currently are in the form of ODEs, and there isn't an easy way to reuse this information in creating new process-algebraic, stochastic models.

In the present work, we developed a methodology to translate simple ODE models into stochastic π-calculus models. We used BioSPI as a platform for representation and stochastic simulation of the process-algebraic model to study its time-dependent behavior. We demonstrated our approach with two case studies dealing with the influence of Raf Kinase Inhibitor Protein (RKIP) on the Extra cellular signal Regulated Kinase (ERK) pathway, and a molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium. We used existing mathematical models for these systems represented with a set of differential equations. We applied our algorithm to extract a set of chemical reactions from the mathematical equations and modeled the new systems using stochastic π-calculus. To verify the accuracy of the models, we simulated it and compared the results with the results obtained by the deterministic models. We found the behaviour of both deterministic and stochastic models to be similar, thus proving the stochastic π-calculus representation to be acceptable for abstraction of biological systems described by the set of ODEs.

Acknowledgements

I am thankful to my supervisor Dr. Sara Kalvala whose help, stimulating suggestions and encouragement helped me in all the time of research and writing of this thesis.

I want to thank Department of Computer Science and its staff members for providing all the necessary supports needed to do this research.

My colleagues Ashutosh Trivedi, Nikolaos Papanikolaou and Antony Holmes deserve big thanks for supporting me in all possible ways. It is a great experience working with these highly efficient and enthusiastic people. Their valuable feedback helped me to improve the thesis in many ways.

Especially, I would like to thank my parents for their continuous support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A model is an *abstraction* of a system. An abstraction can be defined as a mapping from a real-world domain to a mathematical domain, highlighting some essential properties of a system and ignoring the complicated ones. Modeling is an important tool in the scientist's tool kit that helps them in trying to understand reality. Accurate modeling of even small part of reality is a non-trivial task. Yet modeling allows us to interact iteratively with reality and test our assumptions to the extent that behaviour of a model matches the behaviour of the real-life system. The fundamental goal of a model is to explain existing data and to predict system behaviour.

1.1 Modeling in Biology

Recent advances in experimental and computational technologies have revolutionized biological sciences. Large projects like the human-genome project have generated massive amounts of data. Though the project is over, the data analysis will take several years. Molecular biology has uncovered many biological facts like gene sequences and protein properties but this is not sufficient for understanding biological systems. Suppose that we have succeeded in recognizing all of the genes in a cell and understood their functionalities. This information is still local and fragmentary and does not tell us about how cell works as a whole. Discovering all the genes and proteins in an organism is important but at the same time there is a need to understand the structure and dynamics of the system. As Kitano points out [Kit02a], though biological systems are said to be *complex systems*, there is difference between them. Complex systems have large numbers of simple and identical components which together produce *complex* behaviours. These components are like black-boxes whose internal structure either does not exist or is ignored. This is

not the case with biological systems where an individual component has an individual internal structure and thus different behaviour from other components. Selective components interact with each other and produce coherent behaviours. Function in a complex system of simple components emerges from properties of the network they form rather than any specific element, whereas function in biological systems can be attributed to a combination of the network and the specific elements involved. The intrinsic complexity of biological systems demands that a system-level understanding should be the primary goal of biology. A combination of experimental and computational modeling can help us better understand biological systems.

1.1.1 Defining system biology

The view discussed above has resulted in a new perspective of looking at biological systems, where we are looking at the structure and dynamics of systems rather than focusing on the molecular level. This approach is known as *system biology*. This term was first coined by Hiroaki Kitano [Kit02b]. He mentioned that a system-level understanding of a biological system can be derived from insight into four different properties. These four properties are the cornerstones of system biology and are represented in the diagram in Figure 1.1.

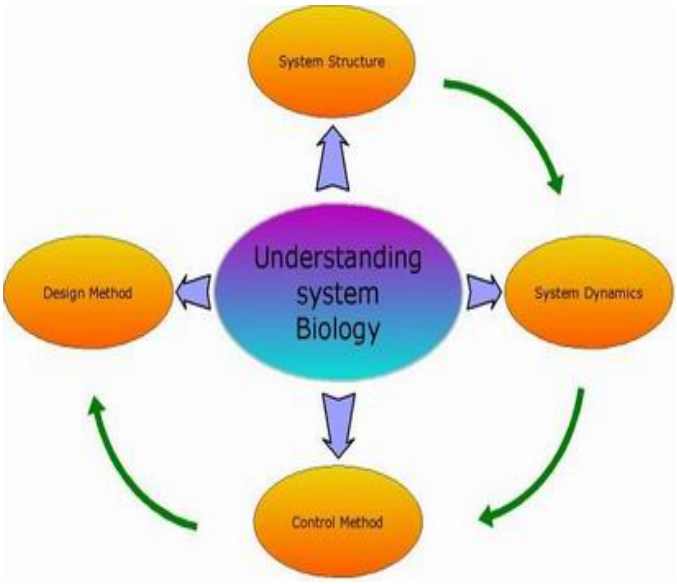

Figure 1.1: Cycle in system biology

System structure includes gene regulation networks and biochemical pathways. It is important to infer the underlying network structure that defines a system. Several

databases have been designed for the collection of entities defining networks. They are a good source of knowledge but many network structures are yet to be identified. This phase allows room for hypothesis based research as shown in Figure 1.2.

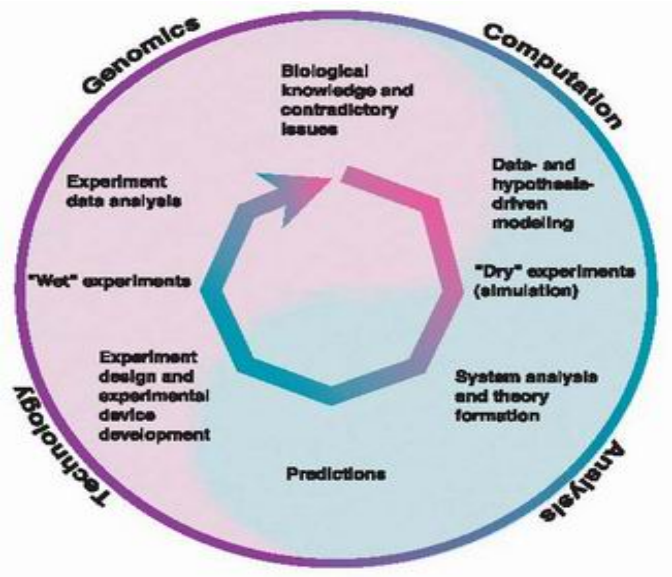

Figure 1.2: Hypothesis driven research. Adapted from [Kit02b]

Once the system structure has been understood, we should be able to investigate the *system dynamics*. System dynamics can give a useful insight into the behaviour of a system over time under various conditions. Analysis of the dynamics of a system requires building a model to describe it. The purpose of the model should be carefully considered and its abstraction level and scope must be defined. We will come to the modeling techniques used in system biology in later sections. This type of analysis has been applied in many biological simulations and has been found to be very useful in predicting the future behaviour of a system.

The third phase of *control methods* emphasises on systematically controlling the malfunctioning elements in a system and identify potential therapeutic targets for treatment of diseases.

If we understand a system well, we should be able to *design* and *create* a new system with some desired properties. This is very important when developing a new drug. When synthesizing a new drug, we expect it to behave in a certain way so that it can cure a disease. Design methods devise strategies to make such drugs or systems. Those strategies are based on principles and knowledge about the system rather than trial-and-error.

1.1.2 Modeling techniques for biology

The computational approach in biology can be divided into two distinct branches: *knowledge discovery* and *simulation based analysis*. Knowledge discovery or the data-mining approach are used to find hidden patterns in biological data and use predictions based on heuristics. These methods are based on statistical theories and linguistic-based approaches and are used heavily in Bio-informatics. On the other hand, simulation based analysis is based on underlying assumptions about a system that can be tested with the experimental data. Simulation based approaches concentrate on the dynamics of a system and compare results with experimental data to check for missing information in the assumptions about the system. As we shall see, the second phase of *system dynamics* emphasises building models to better understand and predict a system. In this section, we discuss various modeling and abstraction approaches used in system biology.

A good scientific abstraction should have four essential properties [Reg02]. First, it should be *relevant* and should be able to capture the essential properties of a system. It should be *computable*, to allow the simulation of dynamic behaviour and provide insight into qualitative and quantitative behaviour of a system. The abstraction should be *understandable* and should clearly explain the domain framework as well as opening new possibilities for thinking about the problem. Finally, it should be *extensible*, allowing the addition of new properties to the framework and should scale to higher levels of organization. We will refer to these properties while discussing efficiency of a modeling technique.

Qualitative models

There are various techniques for modeling biological systems. Firstly there is the *representation* of a system. *Diagrams* provide a good way to reduce complex biological processes into one dimension. These are an easy and elegant way to represent a system and act as natural visual aids when it comes to understanding biological systems. Diagrams have been widely used in biology and will continue to prevail. For this reason, there have been many attempts to formalise the conventions used to draw them. We will discuss some of the methods in later sections. Before we go to the examples, it would be important to mention that diagrams have inspired many computational approaches to look into complexities of biological systems. Topological design, clustering of network components and graph theoretical perspectives are the most prominent amongst them. The major shortcoming with diagrams is that they are only qualitative in nature. They lack quantitative information which is necessary to analyse the dynamics of a system.

A simple way to incorporate dynamics in a diagram is to introduce elementary behaviour into it. *Boolean models* are one such example. Each component of the diagram has two *logical activity states*: *on* or *off*. The state of a variable at time $t+1$ can be determined by looking at the state of the other variables at time t. Boolean diagrams for biology look similar to electronic circuits and are advantageous due to their simplicity and do not require detailed data for modeling. Introduced by Kauffman [Kau93] they have since been used in understanding many biological systems. Due to their simplicity, boolean models suffer from limited predictive power and extensibility. An example of boolean diagram for a biological system is shown in Figure 1.3.

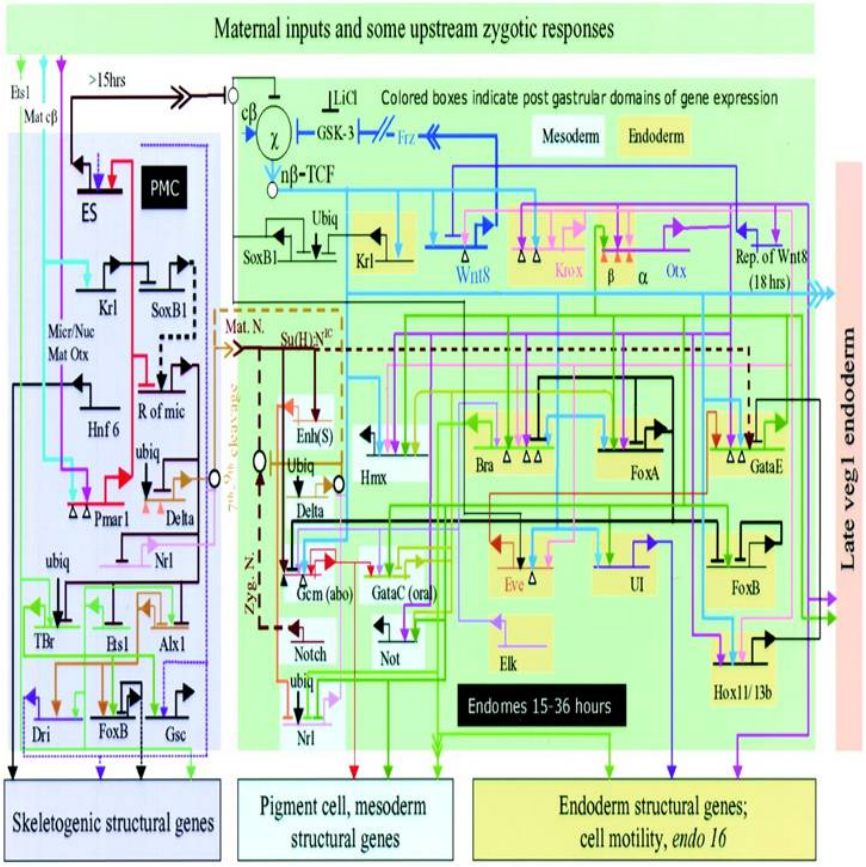

Figure 1.3: Example of biological boolean network. From [Dav03]

Discovery of huge amount of new biological data has resulted in the need for storage strategies. There are many *databases* which are dedicated to storing biological information, storing molecular interaction as well as complete pathways. Example of such databases are KEGG [OGS+99]) and BIND [GIC+01]. These databases have object-oriented schema which provide hierarchical view of molecular entities. Link between entities are defined with the help of *relations*. The databases support query languages

which can be used to extract information about a biological system at different levels of resolution. These databases rely on diagrams to present information about a system to end-user. Most of the time, these diagrams are simple but interactive, like the one from KEGG database shown in Figure 1.4. A mouse-click on an entity reveals more information about it. These databases are an excellent option for organizing, manipulating and visualizing data. However, they suffer from the same problem of lack of dynamic capabilities and have serious qualitative restrictions.

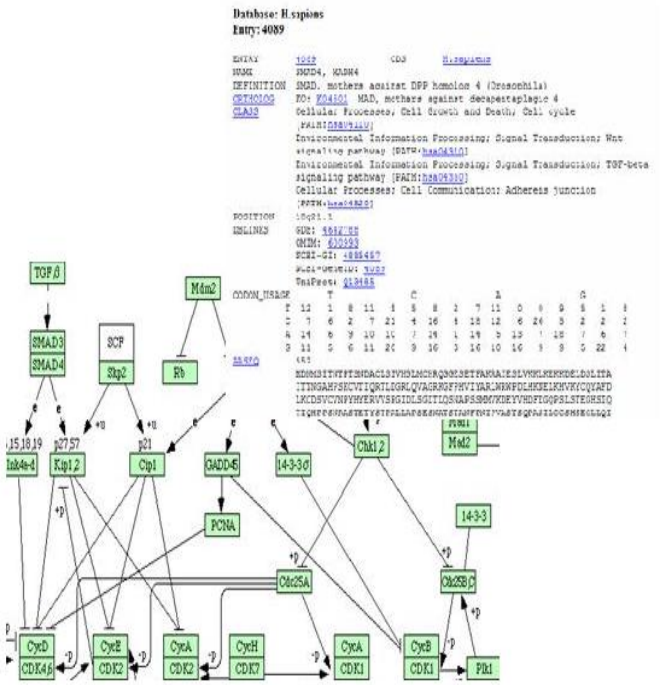

Figure 1.4: Object-oriented based databases of molecular interactions.

These databases use a myriad of different file formats for storing data and that makes the information exchange across them a very difficult task. Kitano foresaw this problem in the early stages of system biology and proposed a standardized way to represent data and models using an XML-based electronic format. This became known as the *System Biology Markup Language(SBML)*. SBML is fast becoming an accepted standard in system biology because of its simplicity and support for many simulation software tools. Although the specification for SBML is huge, we shall illustrate its power using a small example.

The skeleton of a SBML document is shown in Figure 1.5:

```
<model id=001 name="XYZ">
  <listOfCompartments>
    ...
  </listOfCompartments>
  <listOfSpecies>
     ...
  </listOfSpecies>
  <listOfReactions>
     ...
  </listOfReactions>
</model>
```

Figure 1.5: Skeleton of a SBML document

A SBML model consists of a list of compartments(at least one compartment), reacting species and the list of reactions that take place. This information can be coded as shown in Figure 1.6. Species elements can be represented in SBML as shown in Figure 1.7.

SBML works with many simulation softwares like CellDesigner [FK03], [cop05], [KT03] and CellML [AACH03] etc. SBML models essentially represent chemical kinetic theory for simulation of systems. We will discuss kinetic based modeling in later part of this section.

Another diagrammatic representation that can be used for modeling of biological systems is *Petri Net*. Petri nets have been used for the representation,simulation and analysis of biological systems. In short, Petri net diagrams have nodes(circle) and transitions(rectangle). Nodes represent molecular species and transitions represent reactions. Every node has an integer value, known as a *token* associated with it. Tokens represent the number of individual molecules for that species. Transitions(rectangular boxes) have arcs associated with them. An incoming arc into a box represents a reactant and an outgoing one represents a product. The collection of all token numbers at any given point represent the current state of the system. The number of token changes if a reaction happens and this results in a new petri net. Petri nets are essentially a graphical representation of the underlying matrix of the reaction network. A stochastic extension to classical Petri net theory is known as *stochastic Petri net(SPN)*, where transitions fire with an exponentially distributed time delay. Petri nets suffer from the same problem as other modeling techniques discussed. They do not have provision for representing the internal dynamics of an entity. The nodes in a Petri net are black-boxes and there is no knowledge of how they work internally. Biological entities are not like black boxes in nature, so the diagram-based methods discussed above lack in *relevance*.

Another approach to building a qualitative graphical model of a biological systems is to use *Statecharts* [NKH01]. Statecharts can be adapted to an *object-oriented modeling*

```
<listOfReactions>

    <reaction id="R01">
      <listOfReactants>
         <speciesReference species="X0" stoichiometry="1"/>
      </listOfReactants>
      <listOfProducts>
         <speciesReference species="S1" stoichiometry="1"/>
     </listOfProducts>
      <listOfModifiers>
         <modifierSpeciesReference species="M1"/>
     </listOfModifiers>
      <kineticLaw>
         <math xmlns="http://www.w3.org/1998/Math/MathML">
           <apply>
             <times/>
             <ci> k1 </ci>
             <ci> X0 </ci>
             <ci> M1 </ci>
           </apply>
         </math>
         <listOfParameters>
           <parameter id="k1" value="0"/>
         </listOfParameters>
      </kineticLaw>
    </reaction>

    <reaction id="R02">

      ...

    </reaction>

</listOfReactions>
```

Figure 1.6: Presenting reactions in a SBML document

```
<species id="S1" compartment="C1" initialConcentration="2.0"/>
```

Figure 1.7: Defining species and compartment in a SBML document.

framework. They are useful tools for capturing the dynamics of a system because of their rich and clear semantics. Biological entities in statecharts are defined as objects with attributes and variables, and have compartments to show the composition of entities. Relationships between objects are represented by showing communication links between them. Statecharts are able to capture states of a system. Figure 1.9 illustrates this showing a model of T cell. The usefulness of statecharts lies in the fact that they provide a simple visual representation which allows us to zoom into the system, to understand its behaviour at various levels of complexities. Statecharts help us understand how the order of events and the duration of time delays influence the behaviour of the system, and whether contradictory behaviour in output can be explained by recognizing inner states into different clusters [NKH01]. Despite their expressiveness, statecharts have a

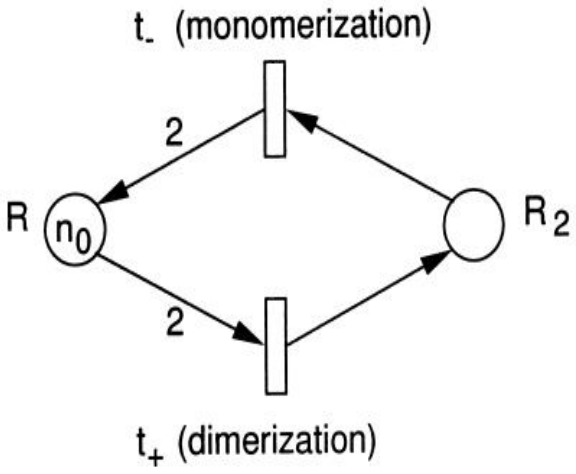

Figure 1.8: A Petri net model of biomolecular dimerization reaction from [PJEG98]
.

serious limitation in handling of quantitative information.

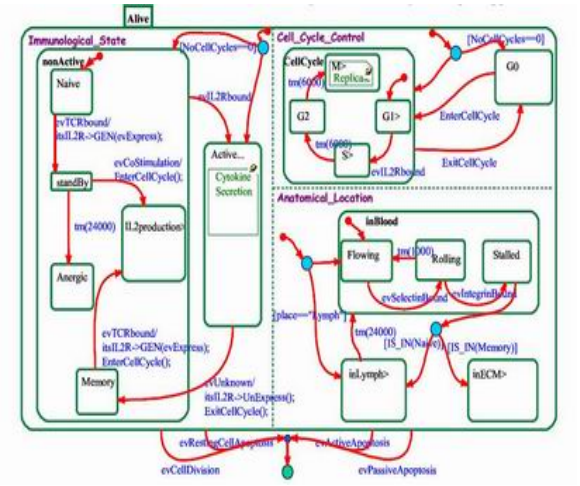

Figure 1.9: Statechart representation of T cell. Adapted from [NKH01].

Quantitative models

In the previous section we discussed modeling techniques which were based on visual representation of systems. Here we will discuss some numerical techniques, also known as *quantitative modeling*. To begin with, we need to define entities and any underlying assumptions. The next step is to derive mathematical equations from this information. This mathematical formulation can be studied for the state of a system at different time

points. As we saw in case of Petri nets, the collection of all the tokens(stoichiometry) defines the state of a system at a given time. These tokens can collectively be seen in the form of a matrix. This representation is one way of incorporating quantitative information into a system.

However, stoichiometric models are difficult to formulate and lack the time-domain in their formulation. Absence of time-domain prevents them from being able to predict the temporal evolution of a system. To compensate for this missing information, these models need to be supplemented with kinetic information about how fast such reactions occur. Usually kinetic models are expressed in terms of ordinary differential equations. Such models are also known as *deterministic models* because a given initial condition determines the behaviour of the underlying system. The velocity of reactions is close to what is measured in experiments and parameter values can be measured from time-series data. Kinetic equation based modeling has enjoyed a most favored status in biology for a long time. Many case-studies have been performed using this method and it is still a very useful tool for modeling biological systems. Differential equation based models can be extended to more complicated form when coupled with other equations describing other processes within the system. Differential equations based modeling has a very good theoretical background and there are very efficient tools for capturing the *dynamics* of a system. Use of differential equations in system biology takes inspiration from *control theory* where we need to control, regulate and coordinate something by means of information feedback to achieve a goal, thus, making the whole system *dynamic* in nature. Various theories like steady-state analysis and bifurcation analysis exist to analyse such systems and they can give great insights into the behaviour of a system and make helpful predictions.

Despite their broad spectrum, differential equation based kinetic models are criticized for their assumption of continuity in molecular concentrations. These models are not suitable for small numbers of participating biological entities that can result in significant random fluctuations at population level and the differential equation based models may fail to capture the randomness of system. If we look at it from the control theory point of view, we find that when it comes to information feedback, control theory assumes that target values are provided to a system designer, but this is not the case in biology where targets are created and changed continuously. Such *self-determined evolution* requires a shift in notion and some different techniques should be used for their abstraction.

Several algorithms exist for *stochastic simulation* that account for randomness in a system. The most popular is the Monte-carlo based Gillespie algorithm [Gil77]. The Gillespie algorithm is based on chemical physics theory and has been found to give exact solution for a network of chemical reactions under certain assumptions. The dynamics

generated by the Gillespie algorithm are different from ones generated by differential equations due to the presence of stochastic fluctuations. Details of the Gillespie algorithm can be found in Chapter 2. The Gillespie algorithm is time-consuming and can take a very long time to find set of solutions as number of reactions in a system increase. Several other algorithms like [GP04], [Gil01], [MF98] also exist which can be used for stochastic simulation of a system.

1.2 Process-algebra as a modeling tool

We introduced many techniques for the modeling of biological systems in the previous section. Major approaches in modeling of such systems are inspired by control theory which assumes that the basic entities of the system are black-boxes whose internal structure does not exist or can be *ignored*. These entities are fundamental objects upon which observations can be made. Karplus [Kar77] defined such systems as black-boxes with inputs and outputs as shown in Figure 1.10. The system (S) receives an input (Excitation, E) and produces an output (Response, R).

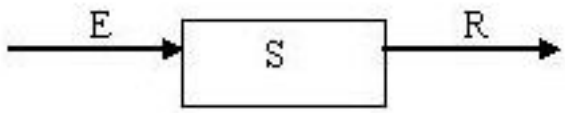

Figure 1.10: Karplus' description of a system as black box.

But in the case of biological systems, the entities have internal structure and behaviours. In fact, the structure of a biological entity defines its function. Let us take as an example of the protein-protein interaction shown in Figure 1.11. Three proteins A,B and C interact with each other. Protein A binds to protein B at a particular location and modifies protein B. Modified B interacts with protein C.

As we can see, there are particular *binding sites* where interactions occur. Once an interaction has occurred, molecules undergo a modification and dissociates to take part in another interaction. Thus, each protein molecule can be made up of several domains each with individual structure and functional part. Different domains interact in a particular way through complementary motifs. An outcome of these interactions is that the molecule may change its shape. A molecule may change its state for example from active to inactive. A molecule may become enabled to interact with other molecule resulting in a chain of events. The classical notion of objects without an internal structure

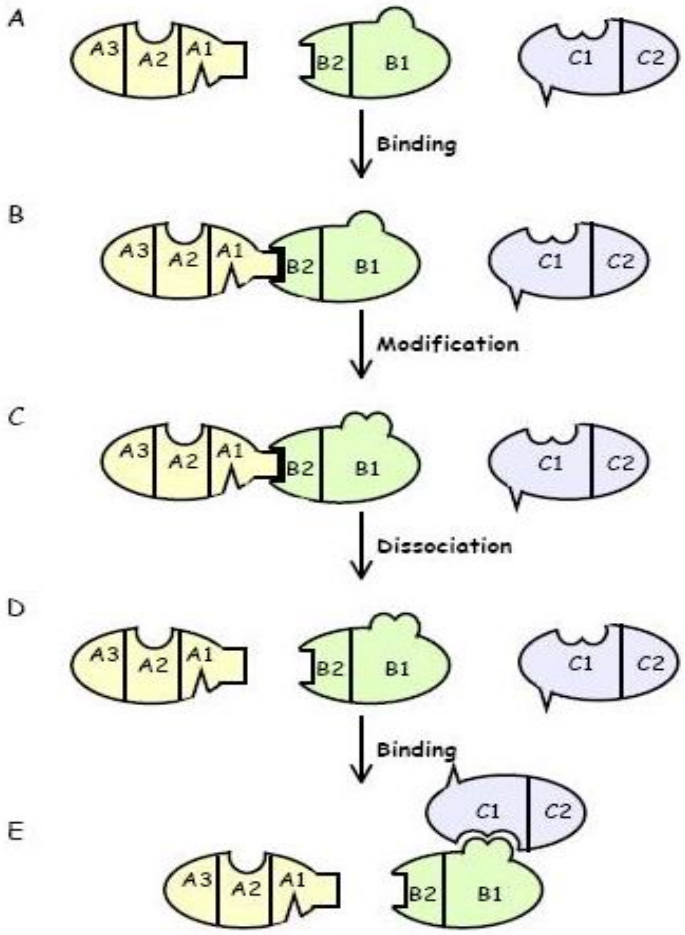

Figure 1.11: Toy example of protein interaction from [Reg02].

does not capture this real-life scenario very well. This is also an example of *self evolving system* which depends on the *localization of complementary interactions*.

We have seen Concurrency theory, especially process-calculi emerging as a tool to model biological systems. The inspiration to use process-algebra for modeling comes from the fact that biological systems are concurrent, heterogeneous and asynchronous in nature. While modeling such systems, biological components are considered as concurrent process and interactions between them as process communication [Car04b]. For example, biomolecular processes are carried out by network of protein molecules interacting with each other.

The protein molecules can be seen as agents, capable of performing computation individually. They exchange information with each other to achieve a common goal. This is the same for concurrent systems, which are made of spatially dispersed,mobile

and communicating computing agents and result in a different computing paradigm, known as *Global Computing*. The process calculi is the most popular formalism to describe and study global computing based applications.

The potential benefits of studying biological systems using this theory includes understanding complex like systems by breaking them into smaller subsystems and analysing their current as well as future behaviour by stochastic simulation. This approach also works for understanding system as a whole [PC04].

There are many variations of process-calculi proposed for the abstraction of biological systems. We will use the π-calculus [Mil99] for this work. Regev and Shapiro proposed representing molecular systems using the π-calculus [Reg02]. It is a name-passing calculus also known as the calculus for *communicating mobile systems*. An stochastic version of this calculus was proposed by Priami [Pri95], which can account for quantitative information in a system.

Coming back to the example in Figure 1.11, the π-calculus can prove to be a suitable tool for modeling a system like this where the internal structure of components should be taken into consideration. We can construct individual entities as *parallel computational processes* and define channels (motifs, where contact takes place) within each process. We can alter these channels after a communication to define a new motif in the molecule. Bio-chemical reactions take place on complementary structural-chemical motifs and also on motif types. The interaction between motifs can be abstracted by defining *communication channels* between corresponding processes. The π-calculus supports two kind of communications. In the first type, a sender process sends a *nil* message to a receiver via a shared communication channel. After the communication, each process may continue as usual or change its state to become a different process with different channels and different behaviour. But in the other type of communication, which is unique to the π-calculus, *self-evolving behaviour* for a system can be achieved. In this type of communication, the sender sends a *message* to receiver. The message contains the names of one or more channels. The receiver can use these channels to decide which process it should interact with next. Passing channel names as messages allows processes to acquire dynamic communication capabilities. These interaction links were not defined a priori but are aquired during the execution of a system by computable processes. This feature is known as *mobility* in the π-calculus. It can also be understood with the help of the diagram in Figures 1.12, 1.13 and 1.14. The diagrams represent a system at different time intervals with different *topological configurations*. In Figure 1.12, the system has four nodes with a defined path for interaction (shown as edges in the graph). The system might undergo transitions and become like that shown in Figure 1.13, where a component D along with its connecting links vanishes from the system. Alternatively,

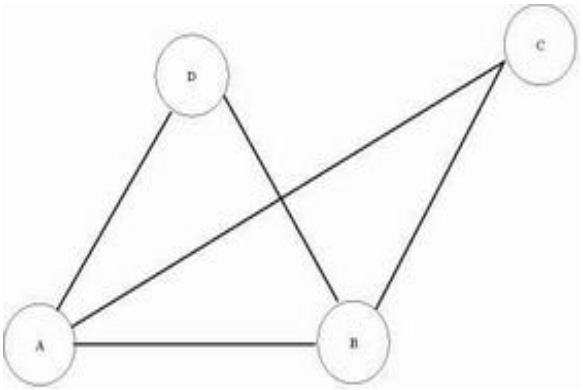

Figure 1.12: Initial configuration. From [Mil99]

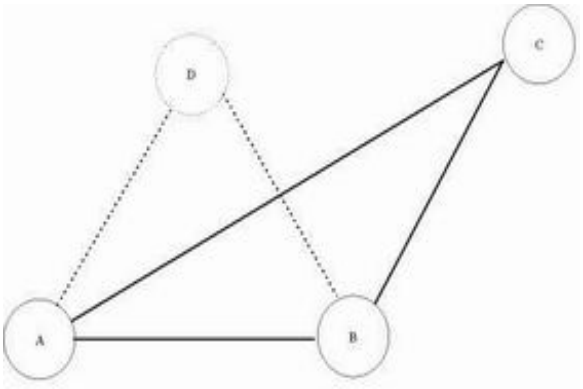

Figure 1.13: Another configuration. From [Mil99]

the system in Figure 1.12 can become like that in Figure 1.14, where node D splits in two nodes D1 and D2 resulting in a new topological configuration for the whole system. The π-calculus is capable of handling such dynamics via its mobility feature. We will discuss formal notations and construction of such systems using the π-calculus in chapter 3.

The abstraction presented by π-calculus is not sufficient for it to be a good modeling tool. It should also come with a way of handling *quantitative information* and should allow dynamic simulation for time-dependent behaviour of a system. The non-deterministic aspect of π-calculus was replaced with a stochastic one by extending the original calculus to *stochastic π-calculus* [Pri95]. We use a modified version of stochastic π-calculus which was incorporated into BioSPI tool [Reg02]. BioSPI is a platform for developing programs in the π-calculus [RSS01]. We use the BioSPI platform for the implementation of our stochastic π-calculus programs. BioSPI takes its inspiration from the Gillespie algorithm for the implementation of its stochastic engine. The semantics of the π-calculus are extended to accommodate reaction rates in channel objects. An

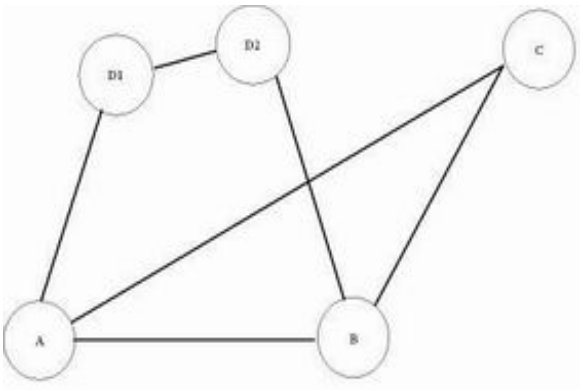

Figure 1.14: Yet another configuration for 1.12. From [Mil99]

explicit clock was introduced in the system which advanced in uneven steps depending on a probability distribution defined by the Gillespie algorithm. The process of selecting objects for communication was also extended from a non-deterministic one to a stochastic one as defined by the Gillespie algorithm. Details about the BioSPI tool can be found in chapter 3.

Representing a system using the π-calculus has another advantage in form of the *notation* used for representing biological systems. Kohn suggested notations to represent biological systems and those notations could be converted into kinetic equations according to a set of guidelines [Koh99]. Though the dynamics of a system can be captured using Kohn's diagram (after converting the representations to equations), the representation itself is static in nature. Another problem with such diagrams is the page-size. If a diagram becomes larger than a page, it becomes less appealing. Cardelli argues that an alternative approach is to devise a textual notation for this purpose which does not have any page-size limit [Car]. He mentions in his paper [Car] that the π-calculus enriched with stochastic semantics is a suitable tool for describing biological activities at molecular level as well as higher level of organization. He further argues that the π-calculus' extension to Ambient calculus [AR04] can incorporate the notion of compartments and complexes in biological systems. The textual notation of π-calculus is not only dynamic but it also provides a direct simulation of the system. The notations can be written in the form of computer programs using the stochastic semantics of the π-calculus and can be passed into BioSPI for execution. BioSPI uses its underlying stochastic machine and produces quantitative data for further numerical or statistical analysis. In this case, we do not need to convert the notations to differential or difference equations and then simulate them. The BioSPI uses the Gillespie algorithm which is a well proven algorithm for the stochastic simulation of biomolecular systems and works well for systems with relatively low number of molecules, a situation where continuous deterministic approaches

may fail.

According to Cardelli, another benefit of using the π-calculus as modeling tool is that it allows *model checking* to be used for validation of models [AR04]. Temporal model checking can be used for analysing if a system can reach a particular state or not, or, if it is necessary to reach one particular state before reaching another state. Quantitative model checkers can help us analyse quantitative values associated with different elements in the system whereas probabilistic model checkers based on Markov chain models can be applied to analyse systems showing probabilistic behaviour.

1.2.1 Current work in field of process-algebra for System biology

Researchers from theoretical computer science community have been developing other tools as well (apart from BioSPI) for modeling of biological systems. We will briefly discuss a few of them in this section.

Cardelli and Phillips [PC04] have developed Stochastic Pi-machine *SPiM*. SPiM is based on a variant of stochastic π-calculus and provides a formal description of how a stochastic π-calculus process can be executed. The simulator for SPiM has been written in OCaml functional language and uses the Gillespie algorithm as basis for stochastic simulation.

Another stochastic simulator is *PEPA* system [GH94] which has been used recently for the stochastic simulation of biological systems [CGH04]. PEPA is a Markovian process-algebra which incorporates activity durations and probabilistic choices. It has been successfully used to determine performance related problems in the design of information systems but PEPA does not the use dynamic name-passing feature (as in the π-calculus) which is important for modeling of biological systems.

Another tool, *PRISM* [MKP01], is a probabilistic model checker and supports three types of models namely, discrete time markov chain,continuous time markov chain and markov decision processes. PRISM supports probabilistic temporal logic for model checking and has been used in number of case studies from biological domain [CVGO05, pria, prib].

1.3 Motivation for the present work

The previous section charts out some of the benefits of using the π-calculus as a modeling tool for system biology. As we have seen, biological systems have traditionally been modeled using differential equations. A vast amount of knowledge about biological systems has been published in form of deterministic differential equations. A plethora of tools have been developed for simulation of such systems. Some of the prominent examples include CellML [AACH03], CellDesigner [FK03] and E-Cell [KT03]. These tools provide complete platform for designing and simulating a biological system, which includes representing a biological system and providing a deterministic as well as stochastic simulation for those systems. These tools have their own notations for representation of a system and are SBML compliant. These tools have benefits of using traditional modeling techniques but lack in the key features of the π-calculus such as self-evolution of a system, being able to capture internal structure of the smallest entities and dynamical representation of a system etc. The main motivation for our work comes from this missing link. If we want to extend these already existing differential equation systems with new found information about them, so that we can look at these systems from a different perspective, we need to first convert those systems from the differential equation domain to the π-calculus domain. We can also get a stochastic simulation for these deterministic systems. Our work concentrates on the automated conversion of deterministic differential equation models to the stochastic π-calculus notations. The resulting notations we obtain are in form of complete programs which could be directly run on the BioSPI system for stochastic simulation. BioSPI produces time-dependent quantitative information for each process involved in the program in the form of a tab-separated file. This file can be used for further mathematical analysis to look into behaviour of a system.

1.4 Our approach

Our approach to convert a differential equation model to a π-calculus model can be divided into four steps.

- In the first step, we implement an ordinary differential equation(ODE) model in Matlab and use its differential equation solver to solve it. A Matlab implementation of a model helps us analyse its dynamic behaviour. The quantitative results are collected for all the species at the steady-state of the system and their time-dependent behaviour are plotted in form of graphs.

- In the second step, a chemical reaction based representation is derived from the system described by ODEs. The new representation has one chemical reaction for each rate constant in the ODE model and can be described as the *reaction-centric* approach. To ensure that the information that we have collected is correct, we implement a stochastic version of the system in Matlab and compare the time-dependent behaviour of the species to the results obtained during the deterministic simulation. We also compare the stochastic results with other software like Copasi [cop05], Stode [GK01] and Dynetica [LYY03] for further assurance.
- When we are satisfied with the results obtained in the above step, we generate another set of chemical reactions which is based on the progressive and decay terms of ODEs. This approach is *reactant-centric* as it describes the whole system with the production and depletion activities of each species. The reactant-centric approach is similar to the reaction-centric approach in terms of information content and forms a basis for the stochastic π-calculus model.
- In the fourth phase, a stochastic π-calculus model is constructed with the set of chemical reactions obtained in the previous step. The model is executed on BioSPI platform and results are compared with the results obtained by deterministic and stochastic simulations in the previous steps.

1.5 Demonstration by two new case-studies

We demonstrate our approach of porting differential equation models to the stochastic π-calculus models with the help of two real-life biological systems. The two case studies are *influence of Raf Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase(ERK) Signaling pathway*[CSK+03] and *A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium* [ML98]. We will discuss the implementation of these systems in detail in chapters 4 and 5 respectively.

1.6 Summary

This chapter started with a brief overview of an emerging discipline, system biology. We discussed various modeling and simulation strategies which are useful for understanding biological systems. We discussed their strengths and weaknesses and introduced the π-calculus as a computational paradigm for modeling biological systems. We presented the motivation for the work in this document and briefly outlined the plan to achieve it.

Chapter 2 presents the mathematical background of our work and Chapter 3 discusses the theoretical and implementational aspects of the π-calculus and its stochastic extension. Chapters 4 and 5 detail the methods used for porting differential equation based models to the stochastic π-calculus models.

Chapter 2

Mathematical Background

2.1 Introduction

Chemical kinetics is the study of speed with which a chemical reaction occurs and the factors that affect the speed. The speed of a reaction is determined by the rate of change in concentrations of reactants and products. Take for example a simple chemical reaction where a reactant A is converted to a product B

$$A \longrightarrow B$$

Here, the velocity,v can be defined as

$$v = -\frac{d[A]}{dt} = \frac{d[B]}{dt} \tag{2.1}$$

When we want to relate experimentally determined initial velocity to concentrations of reactants, we introduce a *rate equation* as

$$v = \frac{d[A]}{dt} = -k[A]^n \tag{2.2}$$

where [A] is the concentration of the reactant A. Concentrations are usually measured in Moles/Litre. The variable 'k' is known as the rate constant and 'n' is an integer which defines the order of a reaction. In general, rate of reaction is proportional to the concentration of each reactant raised to power of its stoichiometry. This is known as *the law of mass-action.*

Chemical reactions can be classified by their order or their molecularity [1]. We saw in equation 2.2, the rate of equation depends on the order n. n is always found by doing experiments and is specific for a reactant. In the above case, the order of reaction with respect to A is 1. *Overall order of a reaction* is found by adding all individual orders in the reaction. In case of reaction like $A+B \longrightarrow C$, the rate equation will be $v = k[A]^1[B]^1$, here, the order of reaction with respect to A and B is still 1, but the *overall order of reaction* becomes 2. The point to be noted is that order of a reaction does not depend on stoichiometry of a reaction as a reaction of third order can have $v = k[A]^1[B]^2$ or $v = k[A]^2[B]^1$.

To analyse the behaviour of a system with respect to time, we write down a set of ordinary differential equations (ODEs) with rate constants ('k' terms, measured in M/sec) and initial concentrations(M/litre) specified. Once it is done, the entire system can be determined by solving these ODEs, either analytically or numerically.

2.2 Michaelis-Menten kinetics

A special case of chemical kinetics is Michaelis-Menten kinetics. It can be understood with help of a simple example from Enzymology that can be expressed as

$$E + S \underset{k_2}{\overset{k_1}{\rightleftharpoons}} ES \xrightarrow{k_3} E + P$$

where Enzyme(E) interacts with Substrate(S) to make an ES complex. The rate constant for formation of ES is defined as k_1 and the rate for dissociation of ES complex is defined as k_2. The conversion of Product(P) from ES occurs at rate k_3. The initial velocity of an enzyme-catalysed reaction is dependent on the present amount of substrate and enzyme concentration. The development of rate equation which allows the velocity of reaction to be correlated with the amount of enzyme is called Michaelis-Menten equation. The formulation of this equation is based on three assumptions. First, the complex ES is in a steady state i.e the concentration of ES remains constant even though many molecules of substrate are converted to the product via ES. The second assumption states, all the molecules of Enzyme are converted to ES complex and there is no free enzyme molecule under saturating conditions. This occurs in case of high concentration of substrates. The third assumption is, if all the enzyme molecules are in ES , then

[1] The number of reactant molecular entities that are involved in the "microscopic chemical event" constituting an elementary reaction.

the rate of formation of product will be maximum. Under these conditions, the rate of formation of product will be maximal.

$$V_{max} = k_3[ES] \tag{2.3}$$

The initial velocity, v_0 is represented as

$$velocity = v_0 = \frac{V_{max}.[S]}{k_m + [S]} \tag{2.4}$$

where

$$k_m = \frac{k_2 + k_3}{k_1} \tag{2.5}$$

k_m is called Michaelis-Menten constant whereas the above velocity equation is known as the Michaelis-Menten equation. In case of substrate population not being very much greater than the number of enzyme molecules present or k_3 not being much lower than other rate constants, the above assumptions fail. It is unclear to what extent these assumptions depend on the absolute particle number in the system. This is a problem for particle based simulations of such systems.

2.3 Lotka-Volterra model

A basic model for interaction of two competing species was proposed by Lotka [Lot25] and, independently, by Volterra [Vol26]. The model can be understood with help of three reactions

$$N_1 \longrightarrow 2N_1$$

$$N_1 + N_2 \longrightarrow 2N_2$$

$$N_2 \longrightarrow 0$$

Here N_1 and N_2 represent Prey and Predator species respectively. The first reaction depicts prey reproduction. The second reaction represents consumption of preys

by predators , resulting in increase in predator population. The third reaction represents death of predators due to natural causes. If we assign rate k_1,k_2 and k_3 to above mentioned reactions respectively, we can formulate ODEs as

$$\frac{d[N_1]}{dt} = k_1[N_1] - k_2[N_1][N_2] \tag{2.6}$$

$$\frac{d[N_2]}{dt} = k_2[N_1][N_2] - k_3[N_2] \tag{2.7}$$

These set of equations can be solved analytically or numerically to understand the evolutionary behaviour of the system. In later sections we will discuss how to solve and analyse ordinary differential equations.

2.4 Concept of equilibrium

An equilibrium is a point defined in a variable space at which the system is at rest. The behaviour of the system does not change at these points and these points are also called stationary or singular points. An equilibrium of a system can be determined by solving set of simultaneous equations by setting the RHS of differential equations to zero i.e in above case $d[N_1]/dt = 0$ and $d[N_2]/dt = 0$ should give the equilibrium points.

$$k_1[N_1] - k_2[N_1][N_2] = 0 \tag{2.8}$$

$$k_2[N_1][N_2] - k_3[N_2] = 0 \tag{2.9}$$

Solving these systems for N_1 and N_2 we have,

$$[N_1] = 0, [N_2] = 0 \tag{2.10}$$

and

$$[N_1] = \frac{k_3}{k_2}, [N_2] = \frac{k_1}{k_2} \tag{2.11}$$

Analysis of equilibrium points can reveal the stability of the system. Stability or instability of a system can be understood by an example of a pendulum. A pendulum is in stable position when it is at rest and is hanging at the bottom-most point in its

trajectory, whereas it is unstable when it is at the top-most point. At the top-most point, it is momentarily at rest but a little perturbation can result into pendulum falling in either of the directions, hence the instability. Study of equilibrium point is important because we assume that most of the natural systems try to reach it to achieve stable state. To study that how a system reaches a stable state, we need to specify initial conditions for our system and assign values to the rate constants.

The solution to differential equations can be studied in various ways and one of the easy ways is to plot the response of a variable against time. Another simple way is to plot two variables against each other as time passes and study their behaviour by following the trajectory. Such plots are known as phase-plane plots and are very effective in studying the behaviour of a system. The above Lotka-Volterra system was simulated with initial parameters, $[N_1] = 10$,$[N_2] = 3$ and rate constants $k_1 = 1$,$k_2 = 0.1$ and $k_3 = 0.1$. The equilibrium points of system are at ($N_1 = 0$, $N_2 = 0$) and ($N_1 = 1$, $N_2 = 10$). The plots are drawn in figure 2.1 and figure 2.2.

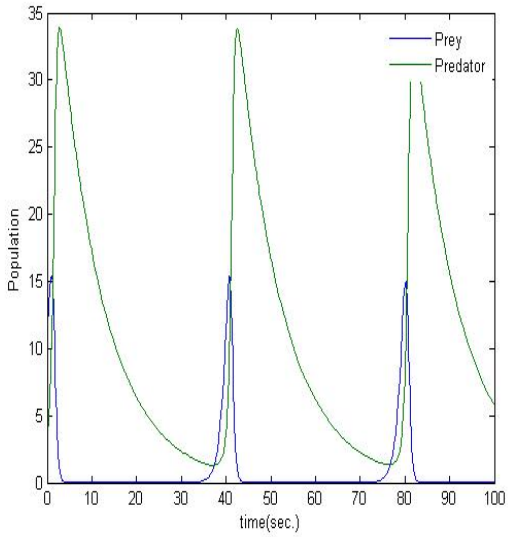

Figure 2.1: Lotka model : Predator and prey

2.5 Numerical solutions to ODEs

In this section, we will discuss general mathematical techniques which can be used for solving a set of ordinary differential equations. Solutions to differential equations cannot be found explicitly in terms of known functions, so an approximate solution for a given

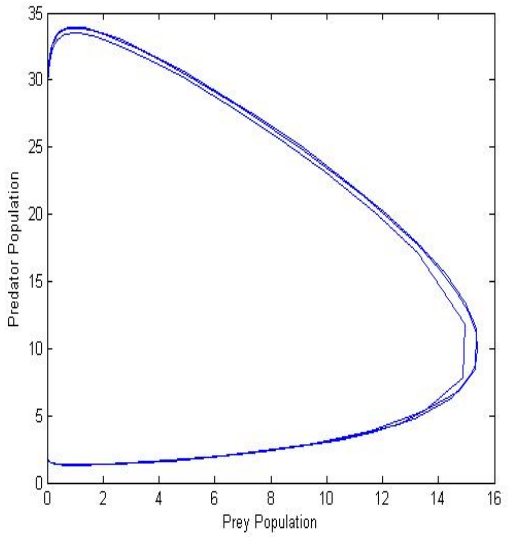

Figure 2.2: Phase plane diagram of Lotka model

data is determined by numerical computations on a computer. The basic idea is to discretize a given differential equation to obtain a system of equations with a finite number of unknowns, which may be solved using a computer to produce an approximate solution. The numerical methods for solving ordinary differential equations are methods of integrating a system of first order differential equations, since higher order ordinary differential equations can be reduced to a set of first order ODE's.

The type of differential equation system which is particularly difficult to deal with is one which exhibits extremes of dynamic behaviour. These systems have periods of time during which the response changes very slowly and sometimes in other periods, possibly very brief, the response is extremely fast. Such systems are known as *stiff systems* of differential equations. The problem with numerical solution to differential equations is the very small step size length δt, which is appropriate when the system is evolving at fast rate but is a limiting factor when the system is changing very slow. This results into excessive amount of computational time. The step size cannot be increased because it would result in poor performance during the high bursts. The solution lies in choosing an adaptive method which moves with a small step size up and down steep slopes and lengthens its pace while encountering plains and plateaux.

2.5.1 Runge-Kutta method

An accurate method of numerically solving differential equations is classical Runge-Kutta method of fourth order. This method requires computation of four auxiliary variables k_1, k_2, k_3, k_4 and then the new value y_{n+1} is computed.

Given initial values x_0, y_0, step-size h and total number of steps N, we can approximate y_{n+1} to the solution $y(x_{n+1})$ at $x_{n+1} = x_0 + (n+1)h$, for $n = 0, 1, ...N-1$ as

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.12}$$

where,

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ x_{n+1} &= x_n + h \end{aligned}$$

Runge-Kutta(RK) is of great practical importance and is an efficient computational algorithm. It is also numerically stable.

E. Fehlberg proposed and developed error control by using two Runge-Kutta methods [Feh68] of different orders to improve from (x_n, y_n) to (x_{n+1}, y_{n+1}). The difference of computed y-value at x_{n+1} gives an error estimate to be used for step-size control. Runge-Kutta-Fehlberg formula has become quite popular and can be given as follows.

Fehlberg's fifth-order RK method

$$y_{n+1} = y_n + \gamma_1 k_1 + ... + \gamma_6 k_6 \tag{2.13}$$

with coefficient vector $\gamma = [\gamma_1 ... \gamma_6]$

$$\gamma = [\frac{16}{135} \quad 0 \quad \frac{6656}{12825} \quad \frac{28561}{56430} \quad \frac{-9}{50} \quad \frac{2}{55}] \tag{2.14}$$

Fehlberg's fourth-order RK method is

$$y^*_{n+1} = y_n + \gamma^*_1 k_1 + ... + \gamma^*_5 k_5 \tag{2.15}$$

with coefficient vector

$$\gamma = [\frac{25}{216} \quad 0 \quad \frac{1408}{2565} \quad \frac{2197}{4104} \quad \frac{-1}{5}] \tag{2.16}$$

In both formulas 6 different function evaluations are used :

$$\begin{aligned}
k_1 &= hf(x_n, y_n) \\
k_2 &= hf(x_n + \frac{1}{4}h, y_n + \frac{1}{4}k_1) \\
k_3 &= hf(x_n + \frac{3}{8}h, y_n + \frac{3}{32}k_1 + \frac{9}{32}k_2) \\
k_4 &= hf(x_n + \frac{12}{13}h, y_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3) \\
k_5 &= hf(x_n + h, y_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4) \\
k_6 &= hf(x_n + \frac{1}{2}h, y_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5)
\end{aligned}$$

Runge-kutta fourth and fifth order method form the basis for Matlab's ordinary differential equation solver, ode45, which we will use for our purpose while discussing the case studies in later chapters.

2.5.2 Implementation of ODE solver in Matlab

Most of the mathematical programming for this project has been done in Matlab. Matlab comes with its own suite of ordinary differential equation solvers which has implementation of many well-known algorithms for solving ODEs. We use 'ode45' solver for our purpose. ode45 is an automatic step-size Runge-Kutta-Fehlberg integration methods. ode45 uses a fourth and fifth order pair of formulas for higher accuracy. Automatic step-size Runge-Kutta algorithms take larger steps where the solution is more slowly changing. Since ode45 uses higher order formulas, it usually takes fewer integration steps and gives a solution more rapidly. Matlab manual recommends that ode45 is the first solver that we should use for our problems. The point to be noted is that ode45 is a solver for non-stiff problems. If we want to solve stiff problems, we should use ode15s,ode23s etc. depending on various factors like crude error tolerance, order of accuracy desired etc.

A typical syntax for invoking an ODE solver in Matlab is

```
[T,Y] = solver(odefun,tspan,y0)
```

or

```
[T,Y] = solver(odefun,tspan,y0,options)
```

where,

- 'solver' can be a predefined routine in Matlab for solving ODEs. Examples of solver are ode45, ode23 etc.
- 'odefun' represents a function-handler in Matlab. A system of differential equation is coded in M-file programming format as a function and it can be called indirectly by means of function-handler. A function-handler can be mapped with the corresponding function by simply specifying

  ```
  odefun = @functionname
  ```

- 'y0' in the argument list of solver is a vector having initial values for parameters in the system.
- 'tspan' defines the time duration for which the system should be run to evaluate the ODEs.
- 'options' is a Matlab structure of optional parameters that change the default integration properties. The various properties that can be set are Relative error tolerance, absolute tolerance, initial step size, maximum step size etc. The list of optional parameters is huge and Matlab provides many ways of adjusting their values for simulation of systems. These properties have pre-defined default values set which can be changed if desired.
- T is a column vector of time points.
- Y is the solution array. Each row in y corresponds to the solution at a time returned in the corresponding row of T.

2.6 Representation of system and derivation of ODEs

Ordinary differential equation models can be derived from a graphical representation of a system. Voit, in his book on computational biochemistry [Voi00] proposed a method to convert biochemical interaction maps into a set of differential equations. He named the mathematical system an *S-system*.

The formulation of a mathematical model from a representation map can be best understood with an example. Consider a reaction, $X_1 \rightarrow X_2$. Production of X_2 is dependent on X_1, whereas X_1 itself is decaying with some rate. Consider a function, V describing the change in X_2 as a function of X_1. Substrate depletion and product formulation can be represented as

$$\frac{dX_1}{dt} = -V(X_1).$$
$$\frac{dX_2}{dt} = V(X_1).$$

For a bi-substrate reaction, like $X_1 + X_2 \rightarrow X_3$, the dynamics of X_3 can be represented as

$$\frac{dX_3}{dt} = V(X_1, X_2).$$

The left hand side of ODEs signifies rate of change in concentration of species X_i, and V is a function of concentration of species that affect the dynamics of X_i. In general, in a system of biochemical reactions, X_i will be a product of one or more reactions and the substrate for one or more other reactions. Thus, the total change in concentration of a species is a combination of production and depletion. So, V can be considered as having two parts V^+ and V^-, depicting production and depletion functions respectively. In general, the change in concentration for a species X_i can be represented as

$$\frac{dX_i}{dt} = V_i^+(X_1, X_2, ..., X_n) - V_i^-(X_1, X_2, ..., X_n)$$

Note that the negative term of the above equation represents the depletion and the positive term represents the production for X_i. Consider another example shown in figure 2.3, where there is a constant influx of X_1 into the system and degradation of X_1 depends on X_1 itself and also on the enzyme,X_3. The system can be described as follows

$$\frac{dX_1}{dt} = \alpha - V_1^-(X_1, X_3).$$

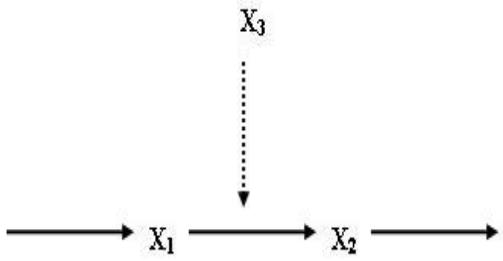

Figure 2.3: Conversion of X_1 to X_2, catalyzed by X_3. From [Voi00]

$$\frac{dX_2}{dt} = V_1^-(X_1, X_3) - V_2^-(X_2)$$

To analyse the system , we need to determine what V_1^- and V_2^- are. Voit claims in his book, that nobody know the answer to this question but numerous considerations of properties and dynamic reactions of biochemical systems suggest that a representation for a process V_i^+ or V_i^- is given by a product of power-law functions of those and only those variables that affect this process; the product is further multiplied by a rate constant that determines the speed of the process. The power-law representation of the processes $V_1^-(X_1, X_3)$ and $V_2^-(X_2)$ are $\beta X_1^a X_3^b$ and γX_2^c respectively. α and β are rate constants and a,b,c represent number of same type of processes participating, or in terms of chemical reactions, a,b,c represent stoichiometry of a chemical reaction. To generalize this model, production rate constants are denoted as α_i and degradation rate constants as β_i. In these production term, the power is called g and in the degradation, it is called h. So, in the above mentioned example, a,b and c can be replaced with h_{11},h_{13} and h_{22} respectively. The first index represents the reaction and the second index represents the species. The parameters α_i and g_{ij} are always used for production terms and β_i and h_{ij} are always used for degradation terms. So,

$$V_i^+(X_1, X_2, ..., X_n, X_{n+1}, ..., X_{n+m}) = \alpha_i X_1^{g_{i1}} X_2^{g_{i2}} ... X_n^{g_{in}}, X_{n+1}^{g_{i,n+1}} ... X_{n+m}^{g_{i,n+m}}$$

$$V_i^-(X_1, X_2, ..., X_n, X_{n+1}, ..., X_{n+m}) = \beta_i X_1^{h_{i1}} X_2^{h_{i2}} ... X_n^{h_{in}}, X_{n+1}^{h_{i,n+1}} ... X_{n+m}^{h_{i,n+m}}$$

The complete system can be represented in the following form

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{n+m} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{n+m} X_j^{h_{ij}} \quad \text{for i = 1,2,....n} \tag{2.17}$$

The system is known as S-system, where S refers to *synergism* and *saturation* of the investigated system as both are fundamental properties of biochemical and biological systems.

As a note, it would be interesting to point out an application of this approach. Many public-domain pathway databases (for example KEGG(Kyoto Encyclopedia of Genes and Genomes) [OGS+99]) describe pathway information with the help of interaction maps, as shown in figure 1.4. This representation is common with most of the databases based on the object-oriented design principle. Though, they may not follow the rules for drawing interaction maps, as proposed by Voit, but the idea of deriving differential equation model from such representations remains the same.

2.7 Stochastic Simulation

While deterministic kinetics is good for large scale chemistry, it does not adequately represents the discrete and stochastic behaviour of intracellular biochemical pathways. As Gillespie pointed out in his paper [Gil77], despite the usefulness of differential equation approach for chemical kinetics, there is something left to be desired at *physical* level. First, differential equation approach assumes that the time evolution of a chemically reactive system is both *deterministic* and *continuous*. In reality, the time evolution of a biochemical system is neither deterministic nor continuous. It is not continuous because the molecular population level of a chemical species can be changed only by a discrete integer value and not by some fraction. And second, the time-evolution is not deterministic either because chemical reactions do not take place at regular intervals or in a deterministic fashion. Chemical events take place when a molecule collides with another molecule while moving around randomly, driven by Brownian motion. This results into a new approach of stochastic modeling of chemical systems. There are states associated with both the models which represent the behaviour of system at various time-points. For deterministic systems, a state is a list of concentration of each chemical species and concentrations are continuous whereas in stochastic models, a state is number of molecules of each chemical species that exist in system and it changes discretely, how and when it changes is probabilistic in nature.

2.7.1 Gillespie's Stochastic Simulation Algorithm (SSA)

Gillespie proposed an algorithm for exact stochastic simulation of coupled chemical reactions [Gil77] based on chemical physics. The algorithm was concerned with a well-stirred solution of fixed volume at constant temperature. Note that this is the assumption for the algorithm. The algorithm is based on the fact that molecules are driven by Brownian motion and keep colliding with each other. Some of the collisions result in chemical reac-

tions. Molecules are distributed randomly and uniformly in the solution. The algorithm proposes a way to compute that which reaction will occur next and when it will take place.

Gillespie's algorithm associates a probability a_μ with every reaction and a_μ is calculated as

$$a_\mu = c_\mu \times h_\mu \tag{2.18}$$

where c_μ is stochastic rate constant and h_μ is the 'hazard' associated with each reaction. The factor c_μ can be constant but h_μ needs to be calculated every time for all the reactions.

$$c_\mu = \frac{k_i \prod_{j=1}^{L_\mu} (l_{\mu j}!)}{V^{K_\mu - 1}} \tag{2.19}$$

and

$$h_\mu = \prod_{j=1}^{L_\mu} \begin{pmatrix} Y_j \\ l_{\mu j} \end{pmatrix} \tag{2.20}$$

k_i are the deterministic rate constants for the chemical reactions. Each reaction μ has K_μ participating reactants, of which there are L_μ different types. For each type there are $l_{\mu j}$ identical reactants, thus $K_\mu = \sum_{j=1}^{L_\mu} l_{\mu j}$. The factor Y_j represents the population of reactant j at that time.

Gillespie's algorithm works as follows:

1. The sum of the probabilities a_0 is calculated as

$$a_0 = \sum_{j=1}^{M} a_\mu \tag{2.21}$$

2. A random number r_1 in the interval $[0, 1]$ is generated and used to determine the time at which the next reaction occurs:

$$\delta t = -\ln(r_1/a_0) \tag{2.22}$$

3. A second random number r_2 is generated, such that

$$\sum_{\nu=1}^{\mu-1} \frac{a_\nu}{a_0} \leq r_2 \leq \sum_{\nu=1}^{\mu} \frac{a_\nu}{a_0} \tag{2.23}$$

r_2 is used to determine the next reaction. This is defined to be the next reaction μ. The population is updated according to the reaction chosen and the time is incremented. The process continues till a time-threshold is reached or system attains a steady-state.

2.7.2 Implementation issues with Gillespie's algorithm

Most of the literature on biochemical reaction is dominated by deterministic kinetics. In order to carry out stochastic simulation of such systems, we can use deterministic kinetic rates and convert them to corresponding stochastic kinetic rates as mentioned by Gillespie's algorithm. The other issue that requires attention is the representation of amount of species in both the models. Deterministic models have amount of species represented as Moles/litre whereas in stochastic models, the amount is represented as a discrete integer value. Since, Gillespie's algorithm assumes that reactions are happening in a well-stirred container, we need to know the volume of the container,V, to compute the number of molecules present for a particular species.

First, we discuss how to convert concentration to number of molecules. Let X be a species with amount [X] Moles in a volume of V litres. Then, there are total [X] $\times$ V moles of X which is equal to $A \times [X] \times V$ molecules of X, where $A = 6.023 \times 10^{23}$ is Avogadro's constant.

Now, we look at the relationship between deterministic and stochastic rate constants. We will refer to the formula proposed by Gillespie in the above section and illustrate it for different orders of reaction.

Consider a Zeroth order reaction,

$$0 \xrightarrow{k} X$$

where k is a deterministic rate constant and is usually represented in units of Ms^{-1}. X is produced at a rate of AkV molecules per second. We can simply write the corresponding stochastic rate constant, c as

$$c = A \times V \times k \tag{2.24}$$

For a first order reaction such as

$$X \xrightarrow{k} \ldots$$

the rate of equation is k[X] Moles per second. X is reducing at the rate of $n_A V[X]k$ molecules per second. The stochastic rate law for this reaction is $n_A V[X]c$. So the stochastic rate constant, c for a first order reaction will be

$$c = k \tag{2.25}$$

Similarly for a second order reaction

$$X + Y \xrightarrow{k} \ldots$$

The deterministic rate law is k[X][Y] Moles per second. The deterministic speed of the reaction is A[X][Y]Vk molecules per second. The stochastic rate law for the reaction is $c \times (A[X]V) \times (A[Y]V)$ molecules per second. Equating them, we get

$$c \times (A[X]V) \times (A[Y]V) = A[X][Y]Vk \Rightarrow c = \frac{k}{AV} \tag{2.26}$$

Similarly, we can perform conversions for higher order reactions as well but they are not very often used in stochastic kinetic model. Once we have performed conversions of amount and rate constants, we can use Gillespie's algorithm for stochastic simulation of a chemically reacting system.

We used Gillespie's algorithm for simulation of chemical systems discussed in case-studies chapter. There is a Matlab implementation of Gillespie's algorithm presented in Appendix. For detailed information on Gillespie's algorithm, one can refer to [Gil77].

Gillespie's algorithm has been applied to many *in silico* biological simulations to study stochastic phenomena [AA98]. But Gillespie algorithm takes huge computational time for a system with large number of reactions. Several other algorithms have been suggested to improve the performance. Gillespie himself came with some new algorithms which could be used for this purpose [GP04], [Gil01]. Some other examples for stochastic simulation of biomolecular systems include Next reaction method [GB00] proposed by Gibson and Bruck and stochsim [MF98] algorithm developed by Morton-Firth.

BioSPI's mathematical engine implements Gillespie algorithm for stochastic simulation of stochastic π-calculus models.

2.8 Software tools

We used several mathematical software tools for verification of our results. A brief introduction to these tools is provided in this section.

2.8.1 STODE

Stode [GK01] was developed to enable the automatic parsing and stochastic simulation of a biochemical system initially described by a set of ordinary differential equations. It developed by Kummer group and later became part of a widely used biochemical network simulator, Copasi [cop05]. Stode uses the Gillespie algorithm for simulation of stochastic systems. It takes as input a set of differential equations in its own specified format and extracts the relevant parameters before performing a stochastic simulation of the system. It shows the extracted information in a XML-type format.

2.8.2 COPASI

COPASI [cop05] stands for *Complex Pathway Simulator* and is a widely used tool for simulation and analysis of biochemical networks. COPASI has an interface which lets us create a model of the system by specifying its details. To do so, we need to identify the species and the reactions occurring in the system. The user-interface of COPASI can be used for modeling of a biochemical system and the model can be simulated using deterministic and stochastic methods.

2.8.3 Dynetica

Dynetica [LYY03] is a simulator of dynamic networks to facilitate model building for networks that can be expressed as reaction networks. It program facilitates easy construction of models for genetic networks. Dynetica can perform time-course simulation of a system using deterministic or stochastic algorithms. Dynetica also provides visual representation of each model for interactive manipulation and interrogation.

2.9 Summary

This chapter provides the basic mathematical background needed for our work. The chapter started with an introduction to chemical kinetics and discussed differential equation based modeling in brief. The chapter compares deterministic differential equation based modeling approach with stochastic modeling approach. The chapter later presents Gillespie algorithm and its implementation issues. A brief introduction to some third-party mathematical software programs which we will use for our work, was also presented in the last section. Next chapter discusses the π-calculus and its stochastic variant. It also introduces BioSPI platform that is used for compiling and executing programs written in the π-calculus and its stochastic variant.

Chapter 3

The π-calculus

3.1 Introduction

In this chapter, we provide an introduction to the π-calculus, a calculus for *concurrent communicating processes* [Mil99]. Conventional computing paradigms like Turing machine,register machines and lambda calculus focus on the *computational* behaviour of a machine. The essential activity in these computational models involves reading or writing on a medium or invoking a procedure with parameters. The *communication activity* of a system is not rigorously defined in these paradigms. The π-calculus presents a model of computation where the basic action between different processes is *communication* . Another important aspect of the π-calculus is the behavioural equivalence of two interacting systems. This means specifying how a designed system should behave. A designed system is said to be correct if its behaviour is equivalent to its specifications.

When we discuss communication, we discuss the *topology* of connections in a network. As Milner mentions, physical system has permanent links where virtual systems like world wide web have *symbolic* links. Symbolic links can be created or destroyed on the fly, depending on the situation. Air-traffic control, Global positioning systems (GPS), General Packet Radio Service(GPRS) systems in mobile phones etc. are example of networks which use symbolic links. We use diagrams(figure 1.12, 1.13 and 1.14) from chapter 1 for discussing the topology of a system.

As we see in figure 1.12, there are four processes A, B, C, and D in a system denoted as circles. The communication path in the system is described as a *link* between two nodes. Over the time, a communication path between processes might remain unchanged or could change to something as shown in figure 1.13, where the node D dies and its

connecting links disappear. Another possibility is, the original system of figure 1.12 can *evolve* as shown in figure 1.14, where node D spawns another node D2 and itself becomes D1, resulting in a new topological structure for the system.

This *evolution* in topological configuration of communicating links is known as *mobility* in a system. The π-calculus is a suitable abstraction paradigm for such *mobile systems*. Another important aspect of the π-calculus is the behavioural equivalence of two different systems. Behavioural equivalence can be understood with the following example. Conside a system having a deterministic finite automaton as shown in figure 3.1. According to the classical theory of automata, there exists an equivalent non-

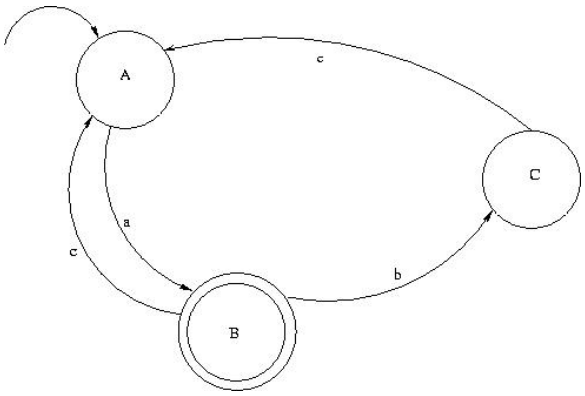

Figure 3.1: Deterministic finite automaton. From [Mil99]

deterministic finite automaton as shown in figure 3.2, which accepts the same regular language as the deterministic finite automaton shown in figure 3.1. Both of the au-

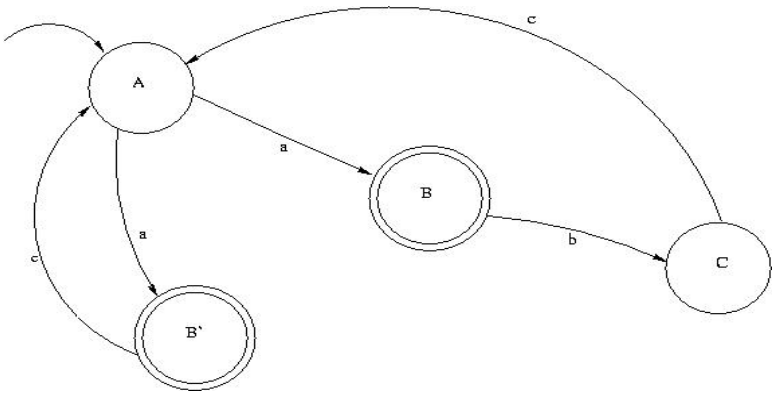

Figure 3.2: Non-Deterministic finite automaton. From [Mil99]

tomata are equivalent because they accept the same regular language, even though their way of execution is different. In case of real-life systems, nondeterminism can not be explained in this way and a nondeterministic automaton can not be equated behaviourally

with a deterministic one [Mil99]. The π-calculus focuses on behavioural equivalence of two interacting systems. This is important to know because unless we know about the similarity or difference in their behaviour, it is difficult to correctly explain what those systems do.

3.2 Constructs in π-calculus

Within the π-calculus, there are two main entities, channels and processes. Processes communicate with each other over channels by a *handshake*. The calculus itself can be divided into three parts, which include

- formally describing the states in a system by a *syntax*,
- having a mechanism to compare two states of a system and determining if they are equivalent by *congruence laws* and
- proposing a set of *reduction rules* to define the change in the state of a system after a communication event.

3.2.1 Communication Action

The starting point is a set of *names*, which are used to name communication channels and the values passed along those channels. Channel names are denoted as a,b,....z etc. whereas processes are represented by P,Q,...and can have the forms shown in table 3.1.

Prefixes	$\alpha ::=$	x ! [ ]	Output nil message on channel x
		x ! $[y_1, y_2 ...., y_n]$	Output $[y_1, y_2 ...., y_n]$ on channel x
		x ? [ ]	Input nil message on x
		x ? $[y_1, y_2 ...., y_n]$	Input $[y_1, y_2 ...., y_n]$ on channel x
Processes	P ::=	0	Nil
		α, P	Prefix
		P + P	Choice
		P|P	Composition
		if x = y then P	Match
		if x $\neq$ y then P	Mismatch
		(new x)P	Restriction
		$A(y_1, y_2, ...y_n)$	Parametric process identifier

Table 3.1: The π-calculus syntax

There are two type of actions associated with a channel. *Output* action, denoted by $x![y_1, y_2...., y_n]$, sends a message as a tuple $[y_1, y_2...., y_n]$ along a channel, named

x, from one process to another process. The complementary *input* action of accepting the tuple $[y_1, y_2...., y_n]$ is performed by another process at channel x and is denoted as $x?[s_1, s_2...., s_n]$. Here $[s_1, s_2...., s_n]$ are placeholders for receiving $[y_1, y_2...., y_n]$. Messages can be empty and the input and output can be represented as $x?[\,]$ and $x![\,]$ respectively. Looking at the representations of the processes in table 3.1, a process can have several forms -

- The empty process $\mathbf{0}$, which cannot perform any further action.
- Prefix action α, P , which can be interpreted as $x![\,], P$ or as some other form defined in table 3.1, can be understood as a process which outputs *nil* message at channel x and becomes process P. While interpreting α, P as $x![y_1], P$, we understand that a message can be received at the channel x, and y_1 is a placeholder for incoming message, the process itself can continue as P.
- A mutually exclusive choice P+Q states that a process can continue either as P or as Q, but not both.
- A composition, represented as P|Q represents parallel concurrent execution of processes P and Q. Processes P and Q can execute independently and are free to communicate with each other.
- A *match*, if $x = y$ then P, states that a process will behave as P if x and y are same, otherwise nothing will happen.
- A *mismatch*, if $x \neq y$ then P, states that a process will behave as P if x and y are not same, otherwise no action will take place.
- *new* is known as *restriction* operator and is used for restricting a name to a local process. The name can be used for interaction within the process but can not be used for interaction between process and its environment.

3.2.2 Congruence laws

The π-calculus introduces *structural congruence laws* to identify processes that represent the same action. The processes might be syntactically different because of linearity of the language but these laws identify processes where it is obvious from their structure that they are the same. For example processes P|Q and Q|P represent the same action but are syntactically different. Both the choice (+ and ;) and composition (|) operations are commutative and associative in nature. The $\mathbf{0}$ process does not perform any action

so it does not have any behaviour and it can safely be discarded while discussing process congruence. There are two types of channels in the π-calculus: free and bounded. Bounded channels are the ones which are in use for communication (input placeholder) or are the restricted ones. As for example, a bounded channel y can have a representation as either x ? [y], P or (new y) P . All other channels are free. Input placeholder channel names are replaced after the communication by the actual channel name received, like , $x?[y], P$ can become $x?[z], (z/yP)$ where channel name y is being replaced by z. A detailed discussion on congruence laws can be found in [Reg02].

3.2.3 Operational Semantics

The operational semantics of the π-calculus defines rules for inter-process communication. As stated earlier, a communication takes place across a shared channel between two processes and it involves message passing. The messages can be nil or can contain names of the channels to be used for further communication by the receiving process. The communication is captured by the semantics of the π-calculus, where the prefixes associated with both the communicating process are eliminated and the remainder represents the new processes. There are five main rules, COMM, PREFIX, PAR, RES and STRUCT for communication and these are summarised in table 3.2.

The α term in table 3.2 is defined in table 3.1. COMM rule states the basic premise of a communication between any two processes by message passing. The tuple(message) contains the names and should be passed on a shared channel. A communication is successful if the received and the sent tuples have the same arity. Once the message has been received, the placeholders for names at the receiving channel are replaced by the actual names received in the message by the renaming operation. PREFIX states occurrence of an action and simply eliminates the prefix associated with a process to give it a new form. Prefixing also induces a sequential order on a communication. Communication under parallel composition(PAR) implies that a communication between two sub-processes of P is independent of the presence of another additional concurrent process Q. Similarly, RES rule states that external restriction on a channel in a process should not affect the ability of internal sub-processes to communicate. The STRUCT rule is important as it suggests that structurally congruent processes undergo similar reductions.

These reduction rules do not specify which communication will occur. They only specify which communications are allowed to occur and what would be the structure of processes involved after a communication has taken place. A system is capable of self-evolution with dynamic message passing among processes and the communication between processes in the system is non-deterministic in nature.

$\frac{a?[x_1,x_2,\ldots,x_n],P \mid a![y_1,y_2,\ldots,y_n],Q}{P\{y_1/x_1,\ldots,y_n/x_n\} \mid Q}$	COMM
$\frac{}{\alpha,P \xrightarrow{\alpha} P}$	PREFIX
$\frac{P \xrightarrow{\alpha} P', BoundName(\alpha) \cup FreeName(Q) = \emptyset}{P \mid Q \xrightarrow{\alpha} P' \mid Q}$	PAR
$\frac{P \xrightarrow{\alpha} P', x \notin \alpha}{(new\ x).P \xrightarrow{\alpha} (new\ x).P'}$	RES
$\frac{P' \cong P, P \xrightarrow{\alpha} Q, Q' \cong Q}{P' \xrightarrow{\alpha} Q'}$	STRUCT
$\frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$	SUM
$\frac{P \xrightarrow{\alpha} P'}{\text{if } x=x \text{ then } P \xrightarrow{\alpha} P'}$	MATCH
$\frac{P \xrightarrow{\alpha} P', x \neq y}{\text{if } x \neq y \text{then } P \xrightarrow{\alpha} P'}$	MISMATCH

Table 3.2: Operational semantics in the π-calculus.

3.3 Stochastic π-calculus

The original framework for the π-calculus is non-deterministic in nature and all the communications in a system are equally likely to occur. But in case of biomolecular systems, a reaction occurs with a particular rate at a random time. To use the π-calculus as a modeling tool for biomolecular systems, the existing calculus needs to be extended to accommodate the *quantitative* information. Chemical reactions are modeled by communication between two processes in the π-calculus. Processes represent molecules. The communication mechanism in the π-calculus should accommodate two features to correctly represent a chemical reaction, the reaction rate and the time for its occurrence. We discussed in section 2.7 about the need for stochastic simulation of a biomolecular system and described the Gillespie algorithm in section 2.7.1 which is an algorithm for the exact stochastic simulation of coupled chemical reactions. We now introduce an extension of the π-calculus to adopt the Gillespie algorithm in its framework [Reg02]. This approach has two benefits. First, the π-calculus abstraction allows representation of a chemical reaction between different molecules having complementary motifs by communication across different processes over the specified channels. Second, by handling channel objects according to specifications by the Gillespie algorithm, we can implement Gillespie's stochastic framework in the existing π-calculus. There are many versions of stochas-

tic π-calculus available [Pri95, PC04, PRSS01], but we will concentrate on the version presented in [Reg02] because it is specifically adapted for the modeling of biomolecular systems.

In order to extend the π-calculus to accommodate the Gillespie algorithm, a few changes are required in the existing framework. Every channel has an associated *base rate*. The *actual rate* for a channel is computed according to its base rate and the number of processes offering to communicate over that channel. An explicit clock is introduced for time-evolution of the system which advances in variable steps by Gillespie algorithm according to the actual channel rates at each state of the system. The choice for communications in a system depends on stochastic selection of channels and is governed by conditions proposed by Gillespie algorithm 2.7. Rest of the syntax remains identical to the non-stochastic version of calculus.

There are different types of elementary reactions which use different kind of defined channels. They can be summarized as follows :

- Asymmetric biomolecular reaction, which involve two reactants from two different species are represented by two different processes using a regular type channel. The actual rate for this type of reaction is $Base\ rate \times \#senders \times \#receivers$
- Symmetric biomolecular reaction involving two reactants from same species are represented by two identical processes using a symmetric type channel. The actual rate for this communication is $Base\ rate \times (\#senders \times \#receivers)/2$.
- Unimolecular reactions which involve only a single reactant can either communicate over a regular public channel with $rate = Base\ rate \times 1 \times receivers$ or over a regular private channel with $rate = Base\ rate \times 1 \times 1$.

We now introduce a formal representation of the stochastic π-calculus. A detailed description can be found in [Reg02]. A *base rate*,r, a non-negative number,is associated with a channel name and it appears in each of communication prefixes, like $(x, r), P$. The processes are assumed to be in *head normal form*. A process P is in head normal form if it is either the null process or

$$P \equiv \sum (\pi_i, r_i), P_i \text{ and } \forall i \neq j.\ sbj(\pi_i) \neq sbj(\pi_j)$$

where sbj(π) denotes the prefix π's output or input link (e.g. if π is a!b(a?b) then sbj(π) is a!(a?)). The actual reaction rates depend on the number of interacting processes, two auxiliary functions, $In, Out : 2^P \times N \to N$, are defined to inductively count the number of receive and send operations on a channel x enabled in a process.

$$
In_x(P) = \begin{cases} 0 & \text{if } P ::= 0 \\ |\{(\pi_i, r_i) | i \in I \wedge sbj(\pi_i) = x?\}| & \text{if } P ::= \sum(\pi_i, r_i), P_i \\ In_x(P_1) + In_x(P_2) & \text{if } P ::= P_1 | P_2 \\ In_x(Q) & \text{if } P ::= new\ z\ .\ Q \text{ and } z \neq x \\ 0 & \text{if } P ::= new\ z\ .\ Q \text{ and } z = x \end{cases}
$$

Out_x is similarly defined, by replacing any occurrence of In with Out and condition $sbj(\pi_i) = x?$ with $sbj(\pi_i) = x![\text{Reg02}]$.

Table 3.3 summarizes the operational semantics of stochastic π-calculus. N represents set of all names. Parameter r_b,r_0 and r_1 represent a channel's base rate, quantity of processes offering to send and quantity of processes offering to receive actions on the channel respectively. r_0 and r_1 are computed compositionally via In_x and Out_x during transitions.

Asymmetric communication$(x \notin H)$**(StochAsym)**
$\ldots + (x!\{z\}, r), Q | (x?\{y\}, r).P + \ldots \overset{x, r_b . 1.1}{\rightarrow} Q | P\{z/y\}$

Symmetric communication$(x \notin H)$**(StochSym)**
$\ldots + (x!\{z\}, r), Q | (x?\{y\}, r).P | \ldots + (x!\{z\}, r), Q | (x?\{y\}, r).P \overset{x, 1/2.r_b . 2.(2-1)}{\rightarrow} Q | P\{z/y\}$

Communication under parallel composition(StochPAR)
$if P \overset{x, r_b . r_0 . r_1}{\rightarrow} P' then\ P | Q \overset{x, r_b . r_0' . r_1'}{\rightarrow} P' | Q, where\ r_0' = r_0 + In_x(Q)\ and\ r_1' = r_1 + Out_x(Q)$

Communication under restriction(StochRes)
$if\ P \overset{x, r_b . r_0 . r_1}{\rightarrow} P'\ then\ new\ x.P \overset{x, r_b . r_0 . r_1}{\rightarrow} new\ x.P'$

Communication and structural congruence(StochStruct)
$if\ Q \cong P, P \overset{x, r_b . r_0 . r_1}{\rightarrow} P',\ and\ P' \cong Q'\ then\ Q \overset{x, r_b . r_0 . r_1}{\rightarrow} Q'$

Table 3.3: Operational semantics of stochastic π-calculus from [Reg02]

Modeling a chemical reaction in stochastic π-calculus

Modeling of a chemical reaction in stochastic π-calculus can be understood with help of a small example. Let A and B be two protein molecules interacting with each other to

produce two different molecules C and D. The initial quantity of A and B before the occurrence of the reaction is n_A and n_B respectively. The base rate for the reaction is k. To model this system, we represent molecules A and B as processes which become processes C and D respectively after the reaction. The chemical reaction between these processes can be represented as a complementary communication taking place over a channel. Let the name of the channel be 'reaction1'. The processes A and B can be defined as

$$\begin{aligned} A &::= reaction1?[\,], C. \\ B &::= reaction1![\,], D. \end{aligned}$$

The actual occurrence of the reaction can be abstracted as:

$$reaction1?[\,], C \quad | \quad reaction1![\,], D \longrightarrow C \quad | \quad D$$

This simple π-calculus module can be extended to its stochastic version by making few changes. First, the reaction channel i.e. reaction1 is assigned a base value of k as reaction1(k) in the beginning of the program. k is a non-negative real number and is computed for a reaction according to the Gillespie's algorithm.Second step is to compute the actual rate for the reaction. The actual rate of the reaction depends on the base value as well as the type of the reaction as described earlier in this section. The actual rate values are calculated at every step and the vary according to time. For this example, the actual rate for reaction between A and B can be calculated as:

$$k \times n_A \times n_B$$

The magnitude of n_A and n_B change after the reaction and the new quantities are used for the calculation of actual rate in the next iteration. The remaining steps are to choose the next communication and time-step according to the Gillespie algorithm. The procedure is repeated for a specified time period or till a system has reached its steady-state.

3.4 BioSPI: A platform for the π-calculus model

BioSPI [Reg02] is an implementation framework for the π-calculus and its stochastic extension. BioSPI is implemented in Flat Concurrent Prolog [Sha87] and the implementation itself is embedded in the Logix System [W.Sb, W.Sa]. Logix system can be installed on Linux, Sgi and Solaris platforms. We used the BioSPI system on the Linux platform. In this section, we will give a brief introduction to writing programs in stochastic π-calculus and executing them on BioSPI.

3.4.1 Representing simple processes

The basic unit of computation is a process. The simplest process is represented as 0 which has no observable behaviour. A process is always declared by a capitalized name in BioSPI.

```
Donothing ::= 0 .
```

`Donothing` is a process which has no behaviour. A system can be composed of different concurrent processes. This composition is represented by PAR(|) operator in the π-calculus. A parallel composition of multiple processes can be represented as

```
BigSystem ::= OneSmallProcess | AnotherSmallProcess .
OneSmallProcess ::= 0 .
AnotherSmallProcess :: = 0 .
```

It represents a system named `Bigsystem`, which is composed of two processes `OneSmallProcess` and `AnotherSmallProcess`. Both the processes operate concurrently within `BigSystem` in a mutually exclusive way.

3.4.2 Communication in BioSPI

Processes communicate with each other on channels, on which they may send messages to each other. There are two types of channels: a public channel which can be used by all the processes in a system or a private channel which is private to a process. Channel are declared with names starting with a lower-case letter.

A *send* action over a channel x is denoted as x![ ] , where [ ] stands for a *nil* message. We can send messages along with channel x as $x![y_1, y_2, ..., y_n]$, where the tuple $(y_1, y_2, ...y_n)$ can have names of channels to be passed on to other process communicating over that channel. Similarly, a *receive* action is denoted as x?[ ] , where a *nil* message is received at the channel x. A receive action with message can be denoted like $x?[y_1, y_2, ..., y_n]$, where the tuple $(y_1, y_2, ..., y_n)$ is a place-holder for actual message and is replaced by the actual message after the communication. Once the communication is over, a process is allowed to retain its state and iterate in the same way as earlier, or, it can change its state and become some other process. This scenario can be represented by the π-calculus code in the following way

```
public(x).

BigSystem ::= OneSmallProcess | AnotherSmallProcess .

OneSmallProcess ::= x ! [ ] , 0 .
AnotherSmallProcess :: = x ? [ ] , 0 .
```

The *comma* infix operator (,) is used to separate a sequence of actions. In the above code, the message sent across the channel x is an empty message. A message with content can be sent as

```
public(x).

BigSystem ::= OneSmallProcess | AnotherSmallProcess | YetAnotherProcess.

OneSmallProcess ::= x ! {z} , 0 .
AnotherSmallProcess :: = x ? {w}, w ! [ ] , 0 .
YetAnotherProcess ::= z ? [ ] , 0 .
```

`OneSmallProcess` sends a channel name z, as a message to `AnotherSmallProcess` through channel x. `AnotherSmallProcess` receives the information and w is replaced with z, making the process enable to communicate with `YetAnotherProcess` at channel z. More than one channel names can be passed in a message as a tuple $x!\{z, p, q\}$.

A private channel is used for communication within the same process. It is known only within that process and its scope is limited. In the π-calculus, a private channel is declared with a *new* keyword but for BioSPI implementation, a private channel is declared during the process declaration, using + operator and is distinct from other public channels even if they have the same name.

```
public(w,x,y,z).
BigSystem ::= OneSmallProcess | AnotherSmallProcess .

OneSmallProcess + x ::= y ! {x} , ( x ! {w} , 0 ; y ! {z} , 0) .
AnotherSmallProcess :: = x ? , 0 .
```

In this code, x is a private channel associated with `OneSmallProcess` and is different from the public channel x in `AnotherSmallProcess`. The operator ';' is called *choice* operator and is used for signifying that a process is capable of offering more than one mutually exclusive communications. This is same as the '+' operator in original π-calculus.

3.4.3 Stochastic programs

BioSPI has an extension for running the stochastic π-calculus programs. The general structure of syntax remains the same as mentioned above but for stochastic programs, the channels are assigned rates. Rates for a channel can be assigned as

```
Rate_1 => 2.0 .

global{
reaction_1(Rate_1).
}.
```

or simply,

```
global{
reaction_1(2.0).
}.
```

global keyword declares that reaction_1 is a public channel.The underlying engine of BioSPI uses the Gillespie algorithm and necessary calculations are performed as discussed in section 3.3.

3.4.4 Operating on BioSPI platform

We will introduce some of the commands important to run the BioSPI system. Programs for BioSPI platform have .cp extension. A file is compiled using a command *c(file name)*. A file named 'nacl.cp' can be compiled as shown in figure 3.4.4. The command

```
@c(nacl)
<1> started
<1> source : /dcs/taps/ritesh/Aspic-release/Tutorial/nacl_example/nacl.cp -
    20050513112458
<1> interpret : export([System / 2, Na / 0, Na_plus /0, Cl / 0, Cl_minus / 0]
    )
<1> file : /dcs/taps/ritesh/Aspic-release/Tutorial/nacl_example/nacl.bin -
    written
<1>  terminated
```

Figure 3.3: Compiling a file in the BioSPI platform

to execute a program is *run*.
run(Goal)
run(Goal,Limit)

Goal specifies the module of the program that we want to execute. The first form executes indefinitely, the second form continues until Limit units of internal time have elapsed. The compiled file can be run as shown in figure 3.4.4. In this case a module *System* is run with parameters (2,2) for 1 unit of time specified by BioSPI system.

```
@run(nacl#"System"(2,2),1)
<2> started
done @1.029526 : seconds = 0
```

Figure 3.4: Executing a program in BioSPI

Another command *record*, resets the session and executes all the goals until Limit. It also records their behaviour on a named file as shown in figure 3.4.4.
record(Goal,File,Limit)

```
@record(nacl#"System"(2,2),nacl_out_1,1)
<3> started
done @1.009714 : seconds = 0
```

Figure 3.5: Recording the output of a program in BioSPI

The file generated by the *record* command needs to be processed to produce the results in a tabular format which is suitable for plotting graphs. The file is passed through

a PERL program 'spi2t', which in turn generates two files. For example,

```
% spi2t  nacl_out_1
```

produces, nacl_out_1.names and nacl_out_1.table. Column 1 of .table file has time information and rest of the columns have quantitative information about all active processes in the system. A small file with .name extension has one line, listing all the column names in .table file and an association of those names with the array columns in the corresponding .table file.

3.5 Summary

The π-calculus, a computational paradigm for concurrent,communicating, mobile systems is introduced in this chapter. We presented the formal representation of the π-calculus and its stochastic variant. We discussed various constructs of stochastic π-calculus which were specifically designed for modeling of biochemical systems. We demonstrated with small code snippets how a π-calculus program can be constructed. We discussed the BioSPI tool and how it can be used for compiling and executing programs written in the π-calculus and its stochastic variant. In the later chapters, we will demonstrate how these concepts can be applied for modeling biochemical systems on the BioSPI platform.

Chapter 4

Method and The Case Study of RKIP on ERK Pathway

4.1 Overview of the methodology proposed

This chapter discusses a method to port an ordinary differential equations(ODE) based model to a stochastic π-calculus model. Chapter 2 and 3 provide the necessary theoretical background needed for our work. We will discuss the method along with the case study of *the influence of Raf Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase(ERK) Signaling pathway*. Before we discuss the case study and the method in detail, a brief outline of the method (independent of the case study) is presented here.

The method to port ODE based system to stochastic π-calculus based model involves several steps.

1. In the first step, an ODE based model, also known as continuous deterministic model is coded on Matlab platform. The model is run with parameters and initial conditions as specified in the publication where the model was first presented. The quantitative results are collected for all the species at the steady-state of the system. Time-dependent behaviour of these species are plotted in form of graphs.

2. In the second step, we derive a chemical equation based representation for the system described by ODEs. This representation has one chemical reaction for each rate constant defined in the ODE model, and can be described as a *reaction-centric* approach. The new model with chemical reactions is coded on Matlab for stochastic simulation according to the Gillespie algorithm. The rate constants

and the initial concentrations of the species are taken from the ODE model and necessary conversions, as proposed by the Gillespie algorithm are performed and incorporated in the new model. The stochastic simulation is run and the results are collected. To check that the information encoded and the results obtained from the stochastic model are correct, we use several third-party software tools namely, Copasi, Stode and Dynetica. These tools facilitate building of biological models for their deterministic and stochastic simulation.

3. When we are satisfied with the results obtained in step 2, we generate another set of chemical reactions which is based on the progressive and decay terms of ODEs. This approach is *reactant-centric* as it describes the whole system with the production and depletion activities of each species. The reactant-centric approach is similar to the reaction-centric approach in terms of information content and forms a basis for the stochastic π-calculus model.

4. A stochastic π-calculus model is constructed with the set of chemical reactions obtained in step 3. The model is executed on BioSPI platform and results are compared with the results obtained in the previous steps.

The ODEs in step 1 were taken directly from the published literature and implemented in Matlab. In the second step, the chemical equation based representation for the system was derived by us by looking at the graphical representation of the system in the publication and the rate constants were obtained from the same source. The Gillespie algorithm for the chemical reactions was implemented by us in Matlab and is presented in Appendix A.3. The derivation of chemical equations in step 3 was automated by a Java program written by us and the details can be found in further section. The π-calculus programs were manually written with the help of output generated by the Java program in step 3.

In later sections, we will discuss the proposed methodology in detail with the help of the RKIP-ERK system.

4.2 Introduction to the case study

The Ras/Raf-1/MEK/ERK pathway is a ubiquitous pathway that conveys mitogenic and differentiation signals from the cell membrane to the nucleus. An important area of research is the role that the kinase inhibitor protein RKIP plays in the behavior of the pathway. The mathematical model for the influence of RKIP on the ERK signaling

pathway was proposed by Cho et.al [CSK+03] and can be explained with the help of the diagram in Figure 4.1.

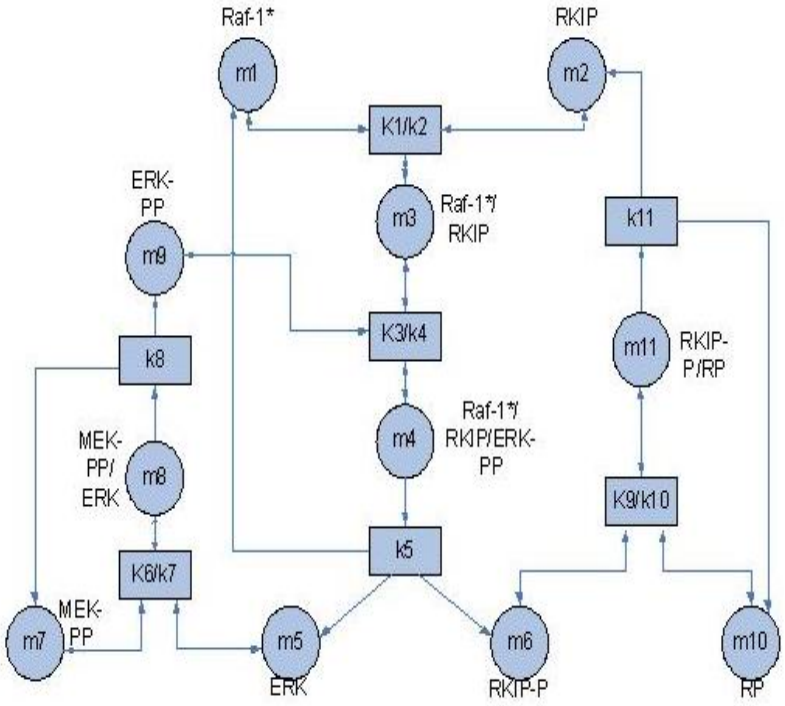

Figure 4.1: Graphical representation of ERK pathway regulated by RKIP [CSK+03]

Cho et.al [CSK+03] explain Figure 4.1 as follows. m_1 denotes the concentration of activated Raf-1 (also known as Raf-1*), m_2 denotes the concentration of RKIP, m_3 denotes the concentration of Raf-1*/RKIP complex, and so on. First, RKIP inhibits Raf-1* to phosphorylate MEK by binding to Raf-1* and forms a Raf-1*/RKIP complex. Free Raf-1* phosphorylates MEK and converts inactive MEK into MEK-PP. MEK-PP binds to ERK and phosphorylates it to active ERK-PP. ERK-PP interacts with Raf-1*/RKIP complex to form another complex, Raf-1*/RKIP/ERK-PP. Then ERK-PP phosporylates RKIP into RKIP-P causing the release of Raf-1* from RKIP-P. ERK is dephosphorylated by Protein Phosphatase 2A(PP2A) and MAPK Phosphates, MKPs. The expression of MKP-1 is transcriptionally induced by ERK. Raf-1* returns to its original state after being released from the Raf-1*/RKIP/ERK-PP complex. The RKIP-phosphotase(RP) is artificially introduced to complete this model by showing dephosphorylation of RKIP-P into the original active state RKIP. After dephosphorylation RKIP binds to Raf-1* and suppresses further phosphorylation and activation of MEK.

4.3 Mathematical formulation of the model

The RKIP-ERK model was mathematically expressed with a set of nonlinear ordinary differential equations (ODEs) based on the mass action enzyme kinetics of the system [CSK+03]. This is a simple model with 11 species and one differential equation for each species. The set of ODEs describing the system are as follows

$$\frac{dm_1(t)}{dt} = -k_1 m_1(t) m_2(t) + k_2 m_3(t) + k_5 m_4(t) \tag{4.1}$$

$$\frac{dm_2(t)}{dt} = -k_1 m_1(t) m_2(t) + k_2 m_3(t) + k_{11} m_{11}(t) \tag{4.2}$$

$$\frac{dm_3(t)}{dt} = k_1 m_1(t) m_2(t) - k_2 m_3(t) - k_3 m_3(t) m_9(t) + k_4 m_4(t) \tag{4.3}$$

$$\frac{dm_4(t)}{dt} = k_3 m_3(t) m_9(t) - k_4 m_4(t) - k_5 m_4(t) \tag{4.4}$$

$$\frac{dm_5(t)}{dt} = k_5 m_4(t) - k_6 m_5(t) m_7(t) + k_7 m_8(t) \tag{4.5}$$

$$\frac{dm_6(t)}{dt} = k_5 m_4(t) - k_9 m_6(t) m_{10}(t) + k_{10} m_{11}(t) \tag{4.6}$$

$$\frac{dm_7(t)}{dt} = -k_6 m_5(t) m_7(t) + k_7 m_8(t) + k_8 m_8(t) \tag{4.7}$$

$$\frac{dm_8(t)}{dt} = k_6 m_5(t) m_7(t) - k_7 m_8(t) - k_8 m_8(t) \tag{4.8}$$

$$\frac{dm_9(t)}{dt} = -k_3 m_3(t) m_9(t) + k_4 m_4(t) + k_8 m_8(t) \tag{4.9}$$

$$\frac{dm_{10}(t)}{dt} = -k_9 m_6(t) m_{10}(t) + k_{10} m_{11}(t) + k_{11} m_{11}(t) \tag{4.10}$$

$$\frac{dm_{11}(t)}{dt} = k_9 m_6(t) m_{10}(t) - k_{10} m_{11}(t) - k_{11} m_{11}(t) \tag{4.11}$$

Parameter estimation for non-linear differential equations is a non-trivial task and the above mentioned work [CSK+03] suggested a novel parameter estimation technique for mathematical modeling. In this approach, authors first discretisized the non-linear ODEs into algebraic difference equations which were linear with respect to the parameters and then solved the transformed linear algebraic difference equations to obtain parameter values at each frozen points. The final parameter values were obtained using regression techniques on those points. For this particular mathematical model the estimated values for the parameters are as mentioned in table 4.1.

Parameter	Estimated Value
k1	0.53
k2	0.0072
k3	0.625
k4	0.00245
k5	0.0315
k6	0.8
k7	0.0075
k8	0.071
k9	0.92
k10	0.00122
k11	0.87

Table 4.1: Deterministic rate constants from [CSK+03]

The system was simulated in the Matlab environment with the help of its ODE solver routine, ode45. The results are discussed in the last section of the chapter.

4.4 Extracting information from deterministic model

The RKIP-ERK system model has the information about the system in form of ODEs. ODEs are higher level mathematical abstraction of biomolecular activities happening at the lower level. The lower level activities can be represented in form of chemical reactions taking place in the system. We discussed in chapter 2, there is a relation between a graphical representation and formulation of differential equations from that representation. Calder et. al have also expressed the RKIP-ERK pathway as producer-consumer relationship while discussing the PEPA model for this system [CGH05]. On our course to chart out a stochastic π-calculus model from the differential equation model, we need to present the mathematical information in a different form.

The π-calculus has two important constituents, processes and channels. We extract chemical equation like representation from ODE model, where species can be represented as processes and their interactions as channels. The graphical representation of RKIP-ERK system as a producer-consumer interpretation, results in 11 chemical equations capturing the interactions in the system. The chemical reactions can be represented in figure 4.2. These reactions provide a lower level abstraction of interaction among molecules in RKIP-ERK system. We pass this system to a stochastic simulator to analyse the dynamics of this system. The implementation of stochastic simulator is discussed in

$$
\begin{aligned}
\text{R1} &: \quad m_1 + m_2 \xrightarrow{k_1} m_3 \\
\text{R2} &: \quad m_3 \xrightarrow{k_2} m_1 + m_2 \\
\text{R3} &: \quad m_3 + m_9 \xrightarrow{k_3} m_4 \\
\text{R4} &: \quad m_4 \xrightarrow{k_4} m_3 + m_9 \\
\text{R5} &: \quad m_4 \xrightarrow{k_5} m_1 + m_5 + m_6 \\
\text{R6} &: \quad m_8 \xrightarrow{k_6} m_5 + m_7 \\
\text{R7} &: \quad m_5 + m_7 \xrightarrow{k_7} m_8 \\
\text{R8} &: \quad m_8 \xrightarrow{k_8} m_7 + m_9 \\
\text{R9} &: \quad m_{11} \xrightarrow{k_9} m_5 + m_{10} \\
\text{R10} &: \quad m_6 + m_{10} \xrightarrow{k_{10}} m_{11} \\
\text{R11} &: \quad m_{11} \xrightarrow{k_{11}} m_2 + m_{10}
\end{aligned}
$$

Figure 4.2: Producer-consumer view of RKIP on ERK.

section 4.5.

4.5 Implementation of stochastic model in Matlab

Section 4.4 summarizes the activities in RKIP-ERK system with the help of chemical reaction representation. This information is enough to construct a stochastic model of this system. To have a stochastic model, we need to perform a few conversions from deterministic domain to stochastic domain. First, we need to convert concentration of species from M/litre to number of molecules/litre and second, we need to convert deterministic rate constants to stochastic rate constants. We use the Gillespie algorithm [Gil77] for stochastic simulation of the system. The choice of the Gillespie algorithm depends on the fact that BioSPI [RSS01] uses the Gillespie algorithm as its stochastic simulation engine and we want to use BioSPI as a platform for implementation of our stochastic π-calculus model. To ensure that we encode correct information in the stochastic π-calculus implementation, we simulate the whole system on our own version of the Gillespie algorithm written in the Matlab environment. A complete stochastic model of this system is given in Appendix A.3.

First, we need to look at the representation of *amount* in both the models. Amount of any reactant in a deterministic model is represented with its *concentration* measured in M i.e. Moles per Liter. Whereas in a stochastic model, the amount is always an integer which represents the total number of molecules of a reactant. In order to carry out the conversion of concentration to number of molecules, we need to fix a constant volume V for the cell. V is measured in liters. For the simulation of our pathway model, we keep V = 1.0e-22 liter.

The continuous deterministic simulation of the system was achieved with the initial population of reactants as mentioned in Table 4.2. The concentration given is in M(Moles/liter) and we need to convert it into number of molecules to achieve stochastic simulation. For a given concentration of a reactant X of [X]M in a volume of V liters, there are [X] $\times$ V moles of reactant X and hence number of molecules, NAV = A $\times$ [X] $\times$ V , where $A = 6.023 \times 10^{23}$ is Avogadro's constant.

The Gillespie algorithm proposes ways to perform the conversion of deterministic rates to stochastic rates. The details of the algorithm can be found in chapter 2. We write the stochastic rate constants for the above 11 equations as shown in table 4.3.

The whole system can be simulated with the Gillespie algorithm for some fixed time

Reactant	Initial concentration(M)
Raf-1*	2.5
RKIP	2.5
Raf-1/RKIP	0
RAF/RKIP/ERK	0
ERK	0
RKIP-P	0
MEK-PP	2.5
MEK-PP/ERK	0
ERK-PP	2.5
RP	3
RKIP-P/RP	0

Table 4.2: Initial concentration of reactants in RKIP-ERK system. From [CSK+03]

duration (100 time units in our case) and can be analysed for its behaviour. The above obtained chemical reactions are represented in form of a matrix(as shown in Figure 4.3) of 11×11 (reactions $\times$ species) dimension. The matrix has entries as $+1$,-1 or 0 according to a species' production, consumption or non-participation in a reaction.

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}
R_1	-1	-1	+1	0	0	0	0	0	0	0	0
R_2	+1	+1	-1	0	0	0	0	0	0	0	0
R_3	0	0	-1	+1	0	0	0	0	-1	0	0
R_4	0	0	+1	-1	0	0	0	0	+1	0	0
R_5	+1	0	0	-1	+1	+1	0	0	0	0	0
R_6	0	0	0	0	+1	0	+1	-1	0	0	0
R_7	0	0	0	0	-1	0	-1	+1	0	0	0
R_8	0	0	0	0	0	0	+1	-1	+1	0	0
R_9	0	0	0	0	+1	0	0	0	0	+1	-1
R_{10}	0	0	0	0	0	-1	0	0	0	-1	+1
R_{11}	0	+1	0	0	0	0	0	0	0	+1	-1

Figure 4.3: Reaction matrix for RKIP-ERK system for Gillespie algorithm

The matrix representation helps in the calculation of propensities of species during the execution of the Gillespie algorithm. We change the amount of species after a reaction with a discrete integer value. In our implementation, that integer value is 1. The amount of a species is either increased by 1 or reduced by 1.

Reaction Number	Deterministic rate	Stochastic rate
k1	0.53	0.0087
k2	0.0072	0.0072
k3	0.625	0.0103
k4	0.00245	0.00245
k5	0.0315	0.0315
k6	0.8	0.0132
k7	0.0075	0.0075
k8	0.071	0.071
k9	0.92	0.0152
k10	0.00122	0.00122
k11	0.87	0.87

Table 4.3: Deterministic and stochastic rates

The stochastic implementation of the system was performed to gather its quantitative behaviour with respect to the Gillespie algorithm. We verify the results and the formulation of chemical reactions with STODE and COPASI softwares. An implementation of the Gillespie algorithm for the complete system is presented in the appendix A.3.

4.5.1 Verification of the model with STODE

The results obtained in the previous section were compared with STODE [GK01]. STODE takes as input a set of differential equations and extracts the relevant parameters before performing a stochastic simulation of a system. Our first goal to verify that we interpreted the ODE based system with correct set of chemical equations was confirmed by this software. Stode has its own format for input of data (set of ODEs and the parameters) and it shows the extracted information in a XML-based file format. Information obtained from Stode helped us verify that the extracted chemical reaction were in synchronization with Stode's approach and we could use this information for designing of the stochastic π-calculus model in BioSPI.

4.5.2 Verification of the model with COPASI

To double-check the information that we gathered was correct, we decided to port the complete model of the system to COPASI and ran the stochastic simulation, so that the results could later be compared with our implementation of the Gillespie algorithm.

It also provided a good opportunity to learn COPASI as a tool. To make a model in COPASI, we needed to identify the species and the reactions occurring in the system. So, instead of giving differential equations as input (as we did for STODE), we gave the chemical reactions as input along with their reaction-rates and the size of the cell. COPASI gathered all the information and generated a set of differential equations for a deterministic simulation of the system. The differential equations obtained were the same as the equations defined in original paper by Cho et.al. This way we confirmed that our approach was correct, first, with the verification by STODE where we gave differential equations as input and generated the chemical equations and then second, with COPASI where we gave chemical equations as input and generated differential equations. COPASI also supports deterministic as well as stochastic simulation of the system. Our system was run for 100 time units for stochastic simulation and the results are presented in form of graphs along with the BioSPI simulation results in appendix B. The results were also used for comparison with our matlab program for accuracy of our code.

4.6 Constructing a stochastic π-calculus model

Once we were satisfied with our stochastic model of RKIP-ERK system, first with our own implementation of the Gillespie algorithm and then with the use of STODE and COPASI, we proceeded to build the stochastic π-calculus model of the system with the help of information obtained in the previous sections.

In the ordinary differential equation representation of the system, there is one equation for each reactant, detailing the involvement of other reactants in the system on the dynamics of that reactant. One way to look at the RKIP-ERK system was, in form of chemical equations discussed in section 4.4. That approach was *reaction-centric* approach where we concentrated on total count of rate constants (k terms) and defined one chemical equation for each rate constant. Another way to look at the RKIP-ERK system, is to follow *reactant-centric* approach where we concentrate on each reactant rather than the rate constants. We derive reactant-centric information from the ODE model itself.

In order to derive reactant-centric representation from the ODEs, we consider two types of reactions for any species. First type is progressive reactions and the other is decay reactions, as suggested by Voit in [Voi00] and discussed in section 2.6. Any reaction that enhances the concentration of a species falls in the category of progressive reactions, whereas any reaction which lowers the the concentration of any species is called a decay reaction. The positive and negative terms of the ODE represent progressive and decay

factors associated with that species. The terms can be treated individually and can be represented in form of individual chemical equations with corresponding kinetic rates. Individual consideration of terms can also be interpreted in a graphical form, as shown in figure 4.1. We interpret each differential equation as a set of chemical reactions in the following way.

For the differential equation 4.1, there are three terms in all. The two terms,$k_2m_3(t)$ and $k_5m_4(t)$ are progressive whereas the third term, $k_1m_1(t)m_2(t)$ is a decay term. The term $k_2m_3(t)$ can be interpreted as, the species m_3 produces m_1 with the rate k_2. It can be represented in form of a chemical equation as

$$m_3 \xrightarrow{k_2} m_1$$

Here, m_1 is being produced at the rate $k_2 \times m_3$. Similarly, the other progressive reaction results in

$$m_4 \xrightarrow{k_5} m_1$$

whereas, the decay reaction in this case is

$$m_1 + m_2 \xrightarrow{k_1} 0 + m_2$$

In this case, m_1 is getting consumed but the consumption rate depends on $[m_1] \times [m_2] \times k_5$. It should be noted that the reactant m_2 does not get affected with this reaction and is just there to *influence* the decay rate of m_1 according to the Gillespie algorithm. The non-decaying nature of m_2 for this reaction is represented by keeping m_2 preserved on the RHS of the chemical equation. m_2, like other reactants in the system has its own set of progressive and decay reactions. Considering it for all the species in the system, their progressive and decay reactions can be summarized in Table 4.4.

The new set of chemical reactions represented in Table 4.4 have the same information as the 11 chemical reactions in section 4.4. If we look carefully, the new set of reactions (listed in Table 4.4)are just rearrangement and combinations of terms defined in the original set of 11 reactions. This new representation depicts a process or reactant-centric approach which is more suitable for creating a stochastic π-calculus model, as it is easier to think about a problem from a process point of view rather than a channel point

Species	Progressive reactions	Decay reactions
m_1	$m_3 \xrightarrow{k_2} m_1$ $m_4 \xrightarrow{k_5} m_1$	$m_1 + m_2 \xrightarrow{k_1} 0 + m_2$
m_2	$m_3 \xrightarrow{k_2} m_2$ $m_{11} \xrightarrow{k_{11}} m_2$	$m_1 + m_2 \xrightarrow{k_1} 0 + m_1$
m_3	$m_1 + m_2 \xrightarrow{k_1} m_3$ $m_4 \xrightarrow{k_4} m_3$	$m_3 \xrightarrow{k_2} 0$ $m_3 + m_9 \xrightarrow{k_3} 0 + m_9$
m_4	$m_3 + m_9 \xrightarrow{k_3} m_4$	$m_4 \xrightarrow{k_4} 0$ $m_4 \xrightarrow{k_5} 0$
m_5	$m_4 \xrightarrow{k_5} m_5$ $m_8 \xrightarrow{k_7} m_5$	$m_5 + m_7 \xrightarrow{k_6} 0 + m_7$
m_6	$m_4 \xrightarrow{k_5} m_6$ $m_{11} \xrightarrow{k_{10}} m_6$	$m_6 + m_{10} \xrightarrow{k_9} 0 + m_{10}$
m_7	$m_8 \xrightarrow{k_7} m_7$ $m_8 \xrightarrow{k_8} m_7$	$m_5 + m_7 \xrightarrow{k_6} 0 + m_5$
m_8	$m_5 + m_7 \xrightarrow{k_6} m_8$	$m_8 \xrightarrow{k_8} 0$ $m_8 \xrightarrow{k_7} 0$
m_9	$m_8 \xrightarrow{k_8} m_9$ $m_4 \xrightarrow{k_4} m_9$	$m_3 + m_9 \xrightarrow{k_3} 0 + m_3$
m_{10}	$m_{11} \xrightarrow{k_{10}} m_{10}$ $m_{11} \xrightarrow{k_{11}} m_{10}$	$m_6 + m_{10} \xrightarrow{k_9} 0 + m_6$
m_{11}	$m_6 + m_{10} \xrightarrow{k_9} m_{11}$	$m_{11} \xrightarrow{k_{10}} 0$ $m_{11} \xrightarrow{k_{11}} 0$

Table 4.4: Progressive and Decay reactions in RKIP on ERK.

of view. We will use these new set of reactions for implementation of the stochastic π-calculus model.

Automation of the process

The process of deriving the information in the reactant-centric form is automated and can be explained in brief as follows.

The automation was achieved using a java program. The program took Matlab file as input and extracted the progressive and decay reactions. The input Matlab file was the same as the one used in section 4.3 for deterministic simulation of the RKIP-ERK system. The code snippet for describing differential equations mentioned in section 4.3 is shown in Figure 4.6.

```
dydt = [
-k1*y(1)*y(2) + k2*y(3) + k5*y(4)                 % m1
-k1*y(1)*y(2) + k2*y(3) + k11*y(11)               % m2
k1*y(1)*y(2) - k2*y(3) - k3*y(3)*y(9) + k4*y(4)   % m3
k3*y(3)*y(9) - k4*y(4) - k5*y(4)                  % m4
k5*y(4) - k6*y(5)*y(7) + k7*y(8)                  % m5
k5*y(4) - k9*y(6)*y(10) + k10*y(11)               % m6
-k6*y(5)*y(7) + k7*y(8) + k8*y(8)                 % m7
k6*y(5)*y(7) - k7*y(8) - k8*y(8)                  % m8
-k3*y(3)*y(9) + k4*y(4) + k8*y(8)                 % m9
-k9*y(6)*y(10) + k10*y(11) + k11*y(11)            % m10
k9*y(6)*y(10) - k10*y(11) - k11*y(11)             % m11
];
```

Figure 4.4: Matlab code snippet for RKIP-ERK pathway

The following steps were required to extract the information detailing progressive and decay reactions from the ODEs. The process is explained with an example.

1. The input Matlab file was parsed to extract the text describing the ODEs. A queue data structure, ODE_collection was created to store this information, with each cell in ODE_collection containing one ODE.
   For example, an entry for the species 'm3', will look like -

   ```
   Ode_collection[3] = "k1*y(1)*y(2) - k2*y(3) - k3*y(3)*y(9) + k4*y(4)  % m3"
   ```

2. While implementing the Matlab code, we took care that every line expressing an ODE ends with a comment as shown in Figure 4.6. The comments helped us map the 'y' terms of the code with the original name of reactants as described in the original model.
   The y terms in ODE_collection were replaced by the original names $m1$,$m2$...etc.

```
Ode_collection[3] = "k1*m1*m2 - k2*m3 - k3*m3*m9 + k4*m4  % m3 "
```

3. Another data structure, Species_collection was created and can be understood with the diagram in Figure 4.6. Each element in Species_collection contained the name

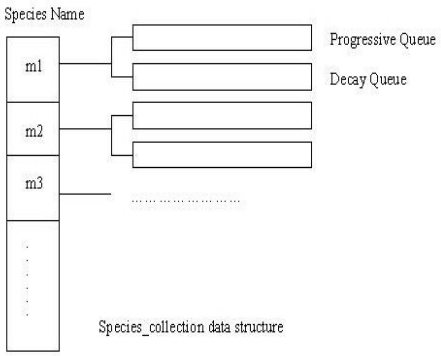

Figure 4.5: Visualization of data structure Species_collection

of the species that it represented and had two queues named "Progressive" and "Decay" to contain the progressive and decay terms of respective ODEs defining that species.

4. For each entry in ODE_collection, extract the ODE expression for that species. Parse the expression into positive and negative terms.

   If a term is positive, add it in the Progressive queue of corresponding entry in Species_collection.

   If a term is negative, add it in the Decay queue of corresponding entry in Species_collection.

   An example of entries for m3 can be seen in figure 4.6

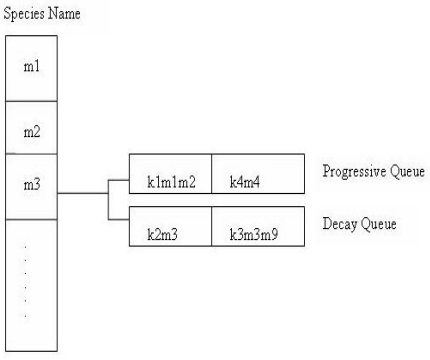

Figure 4.6: Species_collection with entry for species m3

5. For every entry in Species_Collection scan the Progressive queue and display progressive reactions like
   $< m1 >< m2 > |k1| < m3 >, < m4 > |k4| < m3 >$

   Scan the Decay queue and display decay reactions like
   $< m3 > |k2| < 0 >, < m3 >< m9 > |k3| < 0 >< m9 >$

Implementation of stochastic π-calculus model

In this section we will study the implementation of stochastic π-calculus model for the chemical reactions obtained in the previous section. We use BioSPI [RSS01] as the platform for the implementation and analysis of our model. The abstraction of a bio-molecular system is achieved in two steps. First, we create a complete description of the system using the syntax of stochastic π-calculus and then we compute on the abstracted representation to gain insight into its behaviour. BioSPI uses the Gillespie algorithm to achieve stochastic simulation of the system. Gillespie's stochastic framework is implemented within the π-calculus by handling channel objects in the same way as Gillespie handles reaction objects as discussed in section 3.3. The model was constructed with 11 processes and 34 channels, processes representing species(or reactants) and the channels representing the interactions in the system. This information was gathered in the previous section where we generated chemical reactions from the progressive and decay terms of the ODEs. As we can see, the system has only bi-molecular or uni-molecular reactions. A reaction can be represented with the abstraction of species as processes, where they communicate with each other with a complementary communication action performed on a specific channel. For example, a reaction like

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

can be represented as

```
A ::= reaction1 ! [ ] , C.
B ::= reaction1 ? [ ] , true.
```

where *reaction1* is the name of the channel which facilitates the communication between A and B. In the above representation A sends a nil message on the channel

reaction1 and B receives a nil message on the same channel. A gets converted to C, whereas B is consumed in the reaction, becoming a '0' process in terms of the π-calculus. '0' process is denoted with a keyword 'true' in BioSPI . The actual occurrence of the reaction can be represented as

$$\begin{array}{rcl} A & | & B \\ reaction1![\,], C & | & reaction1?[\,], true. \\ & \longrightarrow & \\ C & | & true \end{array}$$

The reaction rate k_1 is associated with the channel *reaction1* and is also called as the base rate of a reaction (See section 3.3 for details). The association between a channel name and a base rate is declared in the beginning of the program and will be discussed shortly hereafter.

BioSPI requires at least two processes for communication to happen. It is fine when we deal with bi-molecular reactions where we have two species communicating to each other. But in the case of unimolecular reactions, a single molecule undergoes a reaction to yield product(s) and we need a communication partner to model this. To overcome this problem, we introduce a dummy process `Timer` which works as an artificial communication counterpart. For example, a species D decays into E and F.

$$D \xrightarrow{k_2} E + F$$

This can be modeled as

```
D ::= reaction2 ? [ ] , E | F.
Timer(reaction2) ::= reaction2 ! [ ], Timer(ch).
```

We can generalize the declaration for 'Timer' process as :

```
Timer(ch) ::= ch ! [ ], Timer(ch).
```

The parametric declaration of Timer Process allows us to use it for different channel parameters for different unimolecular reactions.

Another important point is to generate an appropriate number of processes(or species) for simulation. This is important from the point of view of consumption and production of molecules in the system. Since the stochastic simulation is based on the actual number of molecules, it is important to create appropriate number of copies of a process(or species) before we start with the simulation of the system. This task can be achieved with a small recursive code -

```
Create_M1(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M1 | self .
```

Here `C` is the number of copies we want to make for process `M1`. `C` can be provided externally and the appropriate value for `C` can be calculated by the method discussed in section 4.5 where we obtain the number of molecules from concentration of a species.

While writing the code it is important to follow the conventions defined by BioSPI and that includes the way a process or a channel should be named. As for example, the name of a channel starts with a capital letter as mentioned in its specification. If the name starts with a small letter, it can result into an error. In our program specification, the reactions are represented by channels and all our channels are global in nature. A specific reaction is modeled on a specific channel. The concurrent behaviour of the system is specified in the process module called 'System' with its constituent process-names separated by a '|'. We did not require the mobility feature of the π-calculus for our purpose.

We needed to associate base rates with communication channels to introduce stochastic behaviour in the model. Base rates were calculated according to the Gillespie algorithm for the set of 34 equations derived in Table 4.4 for the modeling of RKIP-ERK system. A correspondence between actual kinetic constants and base rates of channels is summarised in the table 4.5.

We discussed the essential steps required to model the reactions listed in Table 4.4. The dynamics of the procedure can be understood with the help of an example. Consider the reactants M1 and M2 from Table 4.4. M1 and M2 can be represented as processes in BioSPI which participate in certain communication activites represented in form of reactions. Processes M1 and M2 can be defined as

```
M1 ::= reaction_3 ? [] , true ;
       reaction_6 ? [], M1 ;
       reaction_7 ? [], M1 .


M2 ::= reaction_3 ! [] , M2 ;
       reaction_6 ! [], true ;
       reaction_7 ! [], M2 | M3 .
```

reaction_3,reaction_6 and reaction_7 are the communication channels where M1 and M2 can interact with each other and become different processes depending on their specification in the definition. Consider an interaction between M1 and M2 at channel reaction_3.

$$\begin{array}{rcl} M1 & | & M2 \\ reaction\_3?[\,], true & | & reaction\_3![\,], M2. \\ & \longrightarrow & \\ true & | & M2 \end{array}$$

After the communication, M1 becomes 0 but M2 remains as it was. In other words, the *quantity* of M1 was decreased by 1 whereas the *quantity* of M2 remained intact. The above communication can be seen as an abstraction of the chemical reaction $m_1 + m_2 \xrightarrow{k_1} 0 + m_2$, which is the decay reaction of M1 listed in Table 4.4. reaction_3 is the name of the channel and has a base rate associated with it. The base rate is calculated accroding to the Gillespie algorithm and depends on the deterministic rate constants, in this case *k1*. The base rates for each communication channel in the system are listed in Table 4.5.

Similarly, the communication over the channel reaction_6 is

$$
\begin{array}{rcl}
M1 & | & M2 \\
reaction\_6?[\,], M1 & | & reaction\_6![\,], true. \\
\multicolumn{3}{c}{\longrightarrow} \\
M1 & | & true
\end{array}
$$

This communication models the decay reaction for M2, $m_1 + m_2 \xrightarrow{k_1} 0 + m_1$. The communication over channel reaction_7 can be seen as

$$
\begin{array}{rcl}
M1 & | & M2 \\
reaction\_7?[\,], M1 & | & reaction\_7![\,], M2 \;\; | \;\; M3. \\
\multicolumn{3}{c}{\longrightarrow} \\
M1 \;\; | \;\; M2 \;\; | \;\; M3 & &
\end{array}
$$

M1 and M2 remain intact after the communication and a new instance of M3 is spawned, indicating no decay in the population of M1 and M2 and an increase in the quantity of M3. This communication represents the progressive reaction for M3, $m_1 + m_2 \xrightarrow{k_1} m_3$.

The complete specification for the RKIP-ERK pathway can be found in appendix A.1. The model was simulated on the BioSPI platform with the command

```
record(erk\_spi\#"System"(151,151,0,0,0,0,151,0,151,181,0),"erk\_out",100)
```

where `erk_spi` is the name of the program file and `System` is the main process name which need to be run. (151,151,0,0,0,0,151,0,151,181,0) are the number of processes(representing molecules in the true sense) that participate in the simulation. These numbers are obtained by conversion of concentration to number of molecules as discussed in section 4.5. `erk_out` is the output file where simulation results are written and 100 specifies the time units in seconds for which the simulation should be run.

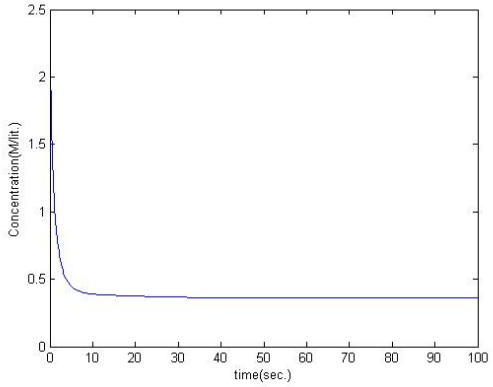

Figure 4.7: Raf-1* (Deterministic simulation)

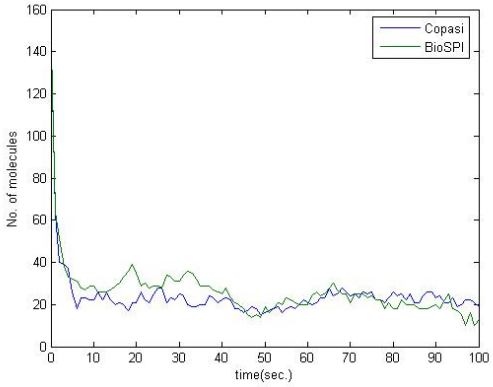

Figure 4.8: Raf-1*(Stochastic simulation)

4.7 Results

The BioSPI model was run for 30 times and the results were averaged over and compared with the ODE model to check if the models was in agreement with each other or not. The results for individual components are obtained and represented in form of graphs in appendix B. The system reached a steady-state after some time. This behaviour is shown by both the models, stochastic one as well as the deterministic one. Table 4.6 summarizes the results obtained. The results obtained by ODE model are listed in second column of the table and are in M/litre unit. Concentrations are converted to number of molecules and are listed in the third column of the table. Column four and five represent the number of molecules obtained by Copasi and BioSPI respectively. As we see, the number of molecules in third, fourth and fifth column are fairly close to each other.

A comparative graphs of deterministic and stochastic model for Raf-1* are shown in Figure 4.4 and 4.5 respectively. Both the graphs exhibit the similar behaviour. The stochastic graph contains the curves obtained by COPASI and BioSPI both. Both the curves are overlapping and display similarity. The deterministic and stochastic graphs for other species in the system can be found in appendix B.

The stochastic graphs for ERK (Figure 4.6 and 4.7), RKPI-P/RP and RKIP-P (figures shown in appendix B) are different from the deterministic ones. There are many horizontal lines present in their stochastic graphs, which indicate that these reactions do not take place very frequently compared to other reactions. The horizontal lines are due

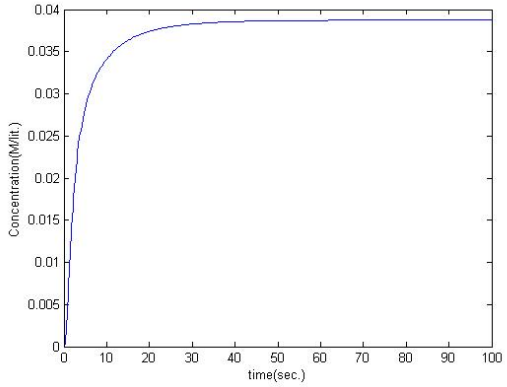

Figure 4.9: ERK (Deterministic simulation)

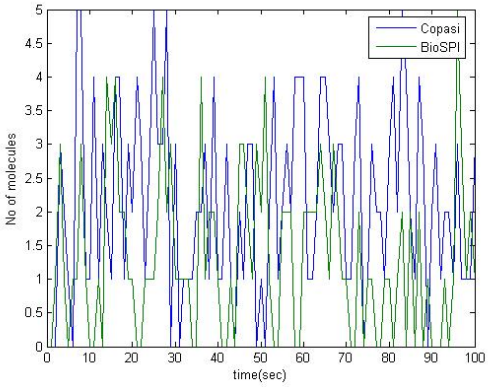

Figure 4.10: ERK (Stochastic simulation)

to the large time-lag between two occurrence of the same reaction. This behaviour cannot be determined by looking at their deterministic model graphs. Looking at the table 4.6, we find that one reason for less frequent occurrence of reactions involving ERK and RKIP-P can be attributed to their low concentrations. RKIP-P/RP is a complex made from RKIP-P, so it also has less frequent involvement among the reactions composing the system. If we just compare the quantities of these reactants in the deterministic and stochastic models, as shown in table 4.6 and also looking at the range of concentration(for deterministic graph) and number of molecules (for stochastic graph), we see that they fluctuate in the same range, which means that quantitatively, both the models are equivalent but the stochastic behaviour indicates about the frequency of reactions as well, which is not evident with the deterministic model. A similar explanation holds true for reactants RP (Figure C) and Raf-1*/RKIP (Figure B) as well. Though the stochastic plots display the similar trend as their deterministic counterparts, there seems to be a gap between trajectories produced by Copasi and BioSPI. It is because the range of y-axis is very small(30 and 50 respectively) for these plots caompared to other plots where the y-values range from 0 to 160. These plots focus only on the specific range to clearly display the dynamics of the ractants.

We see that the graphs obtained from stochastic simulation show enough fluctuations but still maintain the basic behaviour displayed by the graphs in the deterministic model. The study of these fluctuations can be helpful for understanding the system in a better way. The stochastic simulation of the system can also reveal about individual participation of a species or a reaction in that system.

The results produced by the BioSPI model of the system indicate that our approach for conversion of the deterministic model to the stochastic model is in the right direction.

This is confirmed with comparison of quantities of the species at the steady state, and their overall behaviour shown in form of graphs. The similar behaviour displayed by COPASI and BioSPI indicate that both the approaches, reaction-centric and reactant-centric, produce similar behaviour.

4.8 Summary

We developed a methodology to port a ordinary differential equation based biological model to stochastic π-calculus model. We started with a case-study of the influence of Raf Kinase Inhibitor Protein (RKIP) on the Extracellular signal Regulated Kinase(ERK) Signaling pathway and implemented a deterministic model as mentioned in [CSK+03]. On our course to convert this system into a process-algebra model, we first converted them into set of chemical equations with corresponding reaction rates and implemented it on our own version of the Gillespie algorithm. The information was checked for authenticity with the help of software packages STODE and COPASI. The set of chemical equations were automatically rearranged and combined in a way to facilitate easy coding for the stochastic π-calculus model. Channels in the π-calculus model were assigned base rates according to the Gillespie algorithm. The stochastic and deterministic simulation results are presented in the form of graphs in appendix B. In chapter 5, we will discuss another case study of a molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium, and apply the same technique to port the differential equation based model to stochastic π-calculus model.

Channel Number	Deterministic rate	Stochastic rate
Reaction_1	0.0072	0.0072
Reaction_2	0.0315	0.0315
Reaction_3	0.53	0.0087
Reaction_4	0.0072	0.0072
Reaction_5	0.87	0.87
Reaction_6	0.53	0.0087
Reaction_7	0.53	0.0087
Reaction_8	0.00245	0.00245
Reaction_9	0.0072	0.0072
Reaction_10	0.625	0.0103
Reaction_11	0.625	0.0103
Reaction_12	0.00245	0.00245
Reaction_13	0.0315	0.0315
Reaction_14	0.0315	0.0315
Reaction_15	0.0075	0.0075
Reaction_16	0.8	0.0132
Reaction_17	0.0315	0.0315
Reaction_18	0.00122	0.00122
Reaction_19	0.92	0.0152
Reaction_20	0.0075	0.0075
Reaction_21	0.071	0.071
Reaction_22	0.8	0.0132
Reaction_23	0.8	0.0132
Reaction_24	0.071	0.071
Reaction_25	0.0075	0.0075
Reaction_26	0.071	0.071
Reaction_27	0.00245	0.00245
Reaction_28	0.625	0.0103
Reaction_29	0.00122	0.00122
Reaction_30	0.87	0.87
Reaction_31	0.92	0.0152
Reaction_32	0.92	0.0152
Reaction_33	0.00122	0.00122
Reaction_34	0.87	0.87

Table 4.5: Deterministic and stochastic rates

Species	Concen(M/lit.)	#molecules	Copasi	BioSPI
Raf-1* (m1)	0.36	23	19	18
RKIP (m2)	0.28	16.86	15	20
Raf-1*/RKIP (m3)	0.54	32.52	35	45
Raf-1*/RKIP/ERK-PP (m4)	1.6	96.3	98	109
ERK (m5)	0.038	2.28	3	3
RKIP-P (m6)	0.0185	1.11	3	2
MEK-PP (m7)	1.79	107.81	104	95
MEK-PP/ERK (m8)	0.7	42.16	46	40
ERK-PP (m9)	0.159	9.57	5	5
RP (m10)	2.942	177.07	179	163
RKIP-P/RP (m11)	0.057	3.43	1	2

Table 4.6: Comparison table for RKIP-ERK results.

Chapter 5

Case Study of The Dictyostelium Model

5.1 Introduction

A molecular network that produces spontaneous oscillations in excitable cells of Dictyostelium was presented by Michael Laub and William Loomis [ML98]. They simulated the molecular network of underlying adenosine 3',5'-cyclic monophosphate(cAMP) oscillations observed in fields of chemotactic Dictyostelium discoideum cells. The circuit described in the figure below produced the spontaneous oscillations in cAMP observed during the early development of D. discoideum and could account for the synchronization of the cells necessary for chemotaxis and further development. The circuit is named aggregation circuit and it can be understood as follows. The double horizontal line represents the membrane surface of a cell. ACA activates when extracellular cAMP binds to the surface receptor CAR1. Pulses of cAMP are produced after the activation of ACA. CAR1 activates the protein kinase ERK2 that may send signals to ACA. An alternate circuit exists where activation of ACA by CAR1 is not dependent on ERK2. Accumulation of internal cAMP activates kinase PKA by binding to regulatory unit of PKA. ERK2 is inactivated by PKA and no longer inhibits the cAMP phosphodiesterase REGA by phosphorylating it. Activated REG A can hydrolyse internal cAMP. CAR1 is phosphorylated when PKA is activated. Hydrolysis of cAMP by REGA inhibits PKA activity and protein phosphates return CAR1 to high affinity state. The circuit was explained with the help of a set of ordinary differential equations as explained in the next section.

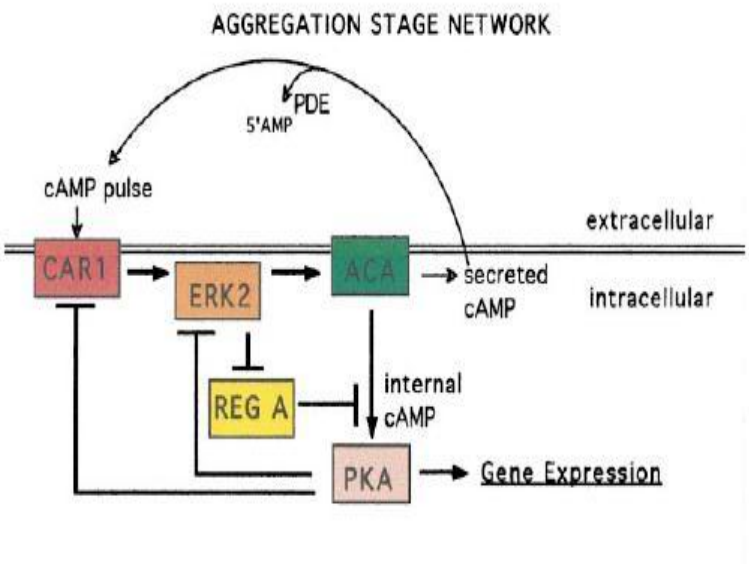

Figure 5.1: Aggregation Stage network [ML98]

5.2 Mathematical formulation of the model

Network model of Dictyostelium was represented in form of a set of ordinary differential equations [ML98]. There are seven differential equations detailing dynamics of seven species present in the system. The equations are as follows

$$\frac{d[ACA]}{dt} = k_1[ERK2] - k_2[ACA] \tag{5.1}$$

$$\frac{d[PKA]}{dt} = k_3[internalcAMP] - k_4[PKA] \tag{5.2}$$

$$\frac{d[ERK2]}{dt} = k_5[CAR1] - k_6[ERK2][PKA] \tag{5.3}$$

$$\frac{d[REGA]}{dt} = k_7 - k_8[REGA][ERK2] \tag{5.4}$$

$$\frac{d[internalcAMP]}{dt} = k_9[ACA] - k_{10}[REGA][internalcAMP] \tag{5.5}$$

$$\frac{d[externalcAMP]}{dt} = k_{11}[ACA] - k_{12}[externalcAMP] \tag{5.6}$$

$$\frac{d[CAR1]}{dt} = k_{13}[externalcAMP] - k_{14}[cAR1][PKA] \tag{5.7}$$

The values for deterministic kinetic constants and their roles are given in table 5.1.

Parameter	Value	Role
k1	1.4	k1 affects activation of ACA (adenylyl cyclase).
k2	0.9	k2 affects inactivation of ACA.
k3	2.5	k3 affects activation of PKA (protein kinase).
k4	1.5	k4 affects inactivation of PKA.
k5	0.6	k5 affects activation of ERK2 (protein kinase).
k6	0.8	k6 affects inactivation of ERK2.
k7	2.0	k7 affects activation of RegA (internal phosphodiesterase).
k8	1.3	k8 affects inactivation of RegA.
k9	0.3	k9 is the proportion of cAMP not secreted.
k10	0.8	k10 affects the breakdown of internal cAMP.
k11	0.7	k11 is the proportion of cAMP secreted.
k12	4.9	k12 is the activity of PdsA (extracellular phosphodiesterase).
k13	18.0	k13 affects activation of CAR1 (cAMP receptor).
k14	1.5	k14 affects inactivation of CAR1.

Table 5.1: Kinetic constants for Dictyostelium deterministic model

The system was simulated in the Matlab environment with the help of its ODE solver routine, ode45. ode45 is automatic step-size Runge-Kutta-Fehlberg integration methods. Automatic step-size Runge-Kutta algorithms take larger steps where the solution is more slowly changing. ode45 uses higher order formulas so it usually takes fewer integration steps and gives a solution more rapidly. The results for this continuous deterministic model are displayed in graphical format in the result section and in appendix C.

5.3 Extracting information from the deterministic model

We follow the same procedure and conventions that we followed for the RKIP-ERK case study in the previous chapter. We analyse each term of a differential equation and classify them as progressive or decay terms according to their sign. This was achieved through the automated process discussed in section 4.6.We extract chemical equation representation of the system from ODEs and list them in table 5.2.

Species	Progressive reactions	Decay reactions
ACA	$ERK2 \xrightarrow{k_1} ACA$	$ACA \xrightarrow{k_2} 0$
PKA	$cAMPi \xrightarrow{k_3} PKA$	$PKA \xrightarrow{k_4} 0$
ERK2	$CAR1 \xrightarrow{k_5} ERK2$	$ERK2 + PKA \xrightarrow{k_6} 0 + PKA$
REGA	$0 \xrightarrow{k_7} REGA$	$ERK2 + REGA \xrightarrow{k_8} 0 + ERK2$
cAMPi	$ACA \xrightarrow{k_9} cAMPi$	$REGA + cAMPi \xrightarrow{k_{10}} 0 + REGA$
cAMPe	$ACA \xrightarrow{k_{11}} cAMPe$	$cAMPe \xrightarrow{k_{12}} 0$
CAR1	$cAMPe \xrightarrow{k_{13}} CAR1$	$PKA + CAR1 \xrightarrow{k_{14}} 0$

Table 5.2: Progressive and decay reactions in Dictyostelium model

5.4 Implementation of the stochastic model

Stochastic modeling of the system was achieved by running the set of chemical equations through our implementation of Gillespie's algorithm. An important thing to note for this system is, there are 14 reactions which represent all individual terms of differential equations. So, for this case, the reaction-centric model is same as the reactant-centric model. To run the system with the Gillespie algorithm, we first convert concentrations of species from Moles/litre to molecules/litre and then we perform conversion of deterministic rate constants to stochastic rate constants.

Deterministic simulation of the system was achieved with the following initial quantity for each reactant

ACA	PKA	ERK2	REGA	cAMPi	cAMPe	CAR1
1.0	1.0	1.0	1.0	1.0	0.1	1.0

Concentrations were converted into number of molecules/litre by multiplying the concentration quantity with the Avogadro's number and the volume of the container. We assumed the fixed volume of the cell to be , V = 1.0e-22 litre. Hence, number of molecules in the container can be calculated as , NAV = $A \times [X] \times V$ where $A = 6.023 \times 10^{23}$ is the Avogadro's constant and [X] is concentration of species in M/litre.

The second step was to convert deterministic rate constants to stochastic rate constants according to the Gillepie algorithm. The converted stochastic rate constants are shown in table 5.3.

Reaction Number	Deterministic rate	Stochastic rate
Reaction-1	1.4	1.4
Reaction-2	0.9	0.9
Reaction-3	2.5	2.5
Reaction-4	1.5	1.5
Reaction-5	0.6	0.6
Reaction-6	0.8	0.0132
Reaction-7	2.0	120.46
Reaction-8	1.3	0.0215
Reaction-9	0.3	0.3
Reaction-10	0.8	0.0132
Reaction-11	0.7	0.7
Reaction-12	4.9	4.9
Reaction-13	18.0	18.0
Reaction-14	1.5	0.0249

Table 5.3: Deterministic and stochastic rates for Dictyostelium model

Once the simulation was performed, we check the results with Dynetica software [LYY03] and then ported the model to the BioSPI platform.

5.5 Verification of system with Dynetica

The above discussed Dictyostelium example has earlier been modeled using the Dynetica software. We came across this example when learning about the Dynetica software. Dynetica model was run for both deterministic and stochastic simulation. Deterministic simulation was achieved with fourth order Runge-Kutta method whereas the stochastic simulation was achieved with the Gillespie algorithm. Our stochastic and deterministic simulation results obtained from our Matlab implementation of the system, were confirmed with the results obtained by Dynetica for accuracy. We present stochastic simulation results from Dynetica and BioSPI models for this case study in the result section.

5.6 Stochastic π-calculus model of the system

To implement the Dictyostelium on the BioSPI system, we used the same approach that we applied for the earlier case-study of RKIP-ERK pathway. The stochastic π calculus program was constructed from the information given in form of progressive and decay reactions listed in table 5.2. Species were modeled as processes and reactions as communication channels. The channels were assigned stochastic rates derived from deterministic rates as listed in table 5.3. Communications in the system were coded with methods discussed in section 4.6, in the same way as the RKIP-ERK pathway model. The system was simulated with arguments :

```
record(dict_spi#"System"(60,60,60,60,60,6,60,1),"dict_out",100)
```

The complete implementation of BioSPI program is divided in two files. First file, rates_dicto.cp has definitions of stochastic reaction rates that can be applied for the communication channels in the system, and the second file dict_spi.cp contains the specification for the whole system. The complete implementation of this case study is presented in Appendix A.2.

5.7 Results

The system was run for 30 times and the results obtained were averaged over before putting in form of graphs. The graphs for species ACA are shown in Figure 5.2 and 5.3. The graphs for all other species in the system are shown in appendix C. The graphs displaying stochastic behaviour have two curves, one generated from the Dynetica software and the other from the BioSPI implementation of the model. We can see that both curves are closely matched and follow the similar pattern, indicating that our approach with the BioSPI model is in agreement with Dynetica's approach of understanding the system. We can also see that the behaviour of individual components match well with the behaviour shown by their counterparts in the deterministic model. The graphs for all the species in the deterministic model display the oscillation property. The similar behaviour can be seen in the stochastic graphs also.

The phase-plane graphs of one species against another are also interesting. Examples of phase plane graphs for ACA vs. REGA are shown in Figure 5.4 and 5.5. Appendix C contains more example of phase-plane plots. For clarity purpose, we present only BioSPI

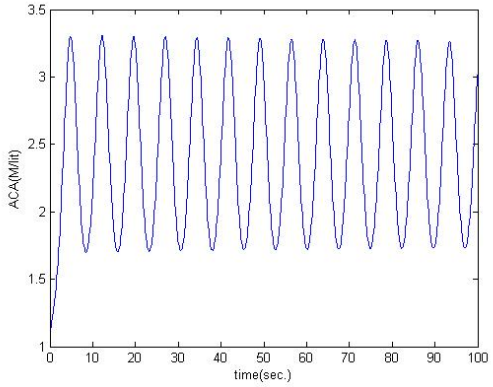

Figure 5.2: ACA (Deterministic simulation)

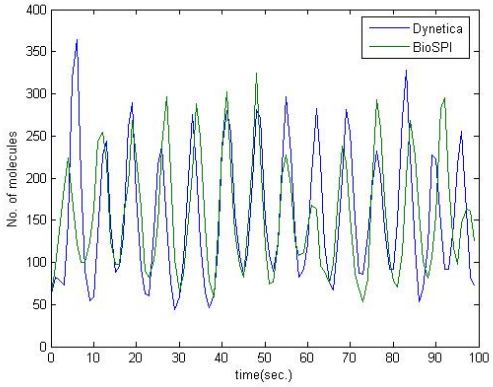

Figure 5.3: ACA (Stochastic simulation)

results while drawing phase-planes. They follow the similar trajectories as described by the deterministic model but also involve lots of deviation while trying to maintain the basic behaviour. Study of such deviations merged with knowledge of molecular biology can provide more insights for such systems. In this example also, we see that the stochastic simulation reveals the non-stationary behaviour of the system which was not evident in the deterministic model. The results obtained in this case-study also indicate that the method developed in the previous chapter, for porting of an ODE based model to a stochastic π-calculus model applies for this case study also and produces the desired results.

5.8 Summary

We presented a new case study of *oscillations in excitable cells of Dictyostelium*. We applied the same methodology that we developed in the previous chapter, where we studied the RKIP-ERK signaling pathway. The results obtained with this case study suggest that our approach is acceptable for porting of ordinary differential equation based models to stochastic π-calculus based models. A more comprehensive knowledge of molecular biology can be useful for interpreting the results produced by these models.

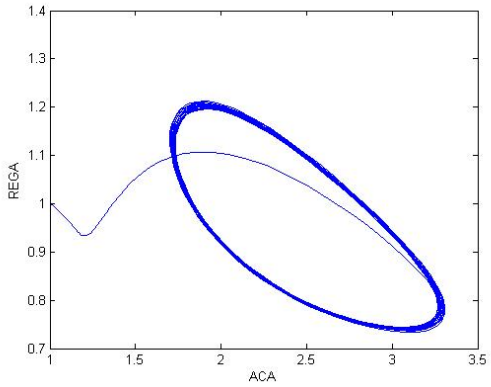

Figure 5.4: Phase-plane plot ACA vs. REGA(Deterministic simulation)

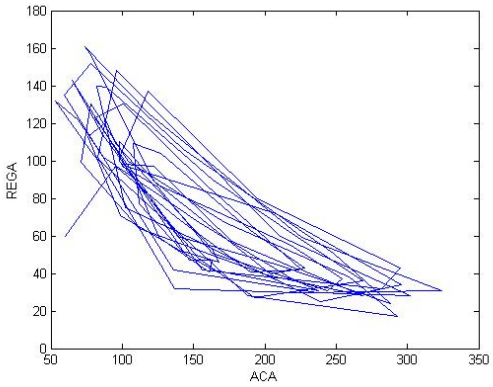

Figure 5.5: Phase plane plot ACA vs. REGA(Stochastic simulation)

Chapter 6

Conclusion

This thesis is a result of a single year's research on modeling and simulation of biological systems, using the stochastic π-calculus. We developed a methodology to port an ordinary differential equation based model of a biological system to a stochastic π-calculus model. In order to conclude the presentation of this work, we will summarize the organisation of each chapter in this thesis and their main achievements. We will discuss some of the open issues that are relevant to our approach and the future directions for the presented work.

6.1 Recapitulation

We started with an introduction to an emerging discipline, system biology. We discussed various qualitative and quantitative modeling techniques used for describing and artificially simulating biological systems. We introduced the π-calculus and its stochastic variant as a modeling tool and discussed its advantages. We then, developed a methodology to extract information from an ordinary differential equation based model of a biological system , to port it to a stochastic π-calculus model. We demonstrated our approach with the help of two real biological case studies, discussed in Chapters 4 and 5. On a per chapter basis, the topics described in this thesis have been :

- **Chapter - 1**
  - introduction to system biology and its goals.
  - qualitative modeling techniques for modeling of biological systems.

- quantitative modeling techniques for dynamic simulation of a biological system.
- comparison between traditional modeling approaches.
- benefits of using π-calculus for modeling of biological systems.

- **Chapter - 2**
  - chemical kinetics and use of ordinary differential equations to model them.
  - Euler and Runge-kutta methods to numerically solve a set of ordinary differential equations and how to use ODE solvers in Matlab.
  - derivation of an ODE based model from a graphical representation of a system.
  - need for stochastic simulation
  - Gillespie's algorithm for exact simulation of coupled chemical reactions
  - practical implementation issues with the Gillespie algorithm.

- **Chapter - 3**
  - introduction to the π-calculus.
  - formal presentation of π-calculus with various rules and semantics explained.
  - stochastic extension of the π-calculus and its relationship with the Gillespie algorithm.
  - BioSPI as a tool for compiling and executing π-calculus based programs, and an introduction to developing small programs on the BioSPI platform.

- **Chapter - 4**
  - overview of the methodology proposed.
  - discussed differential equation based mathematical model of RKIP on ERK pathway.
  - implementation of a stochastic model based on the reaction-centric approach and its simulation using the Gillespie algorithm.
  - implementation of the stochastic model in STODE and COPASI softwares.
  - automated formulation of a reactant-centric model from the original ODE model of RKIP-ERK system.
  - implementation of stochastic π-calculus model for RKIP-ERK system.

- comparison of results for deterministic and stochastic simulations. Graphs also represent that reaction-centric and reactant-centric approaches are the same.

- **Chapter - 5**

  - discussed differential equation based mathematical model of oscillations in excitable cells of Dictyostelium.
  - implementation of a stochastic model based on the reaction-centric approach and its simulation using the Gillespie algorithm.
  - implementation of the stochastic model in Dynetica.
  - automated formulation of a reactant-centric model from the original ODE model of the molecular network describing oscillation in Dictyostelium.
  - implementation of stochastic π-calculus model of the original system described in form of ODEs.
  - comparison of results for deterministic and stochastic simulations.

6.2 Discussion

The complexity of biological systems has motivated researchers from various disciplines to develop computational methods to represent, and simulate these systems in more realistic ways. Several research groups are developing software tools which can facilitate building complex computational models of biological systems. To name a few, CellDesigner [FK03], CellML [AACH03], E-Cell [KT03] etc. are the most popular modeling platforms among researchers. One can build a model by following specifications proposed by them and run the model for deterministic and stochastic simulation. These tools have the benefits of using the traditional modeling techniques but they lack the key features of the π-calculus, such as self-evolution of a system, being able to capture internal structure of smallest entities and dynamical representation of a system using the syntax of the calculus.

Abstract process-algebra based modeling is relatively new for system biology and is gaining popularity among researchers. Recent publications from the theoretical science community have supported this issue [Car04a, Car05, Reg02] and various case studies including the recently concluded ones like transcription in bacteria [Kut05] and gene regulation in Bacteriophage λ [KN04] provide evidence for the correctness of the approach.

Most of the biological models exist in the form of ordinary differential equations. These models are continuously refined and scaled with new discoveries about their systems. Tools like CellDesigner, CellML etc. not only allow building new models but also facilitate porting of old ones, which can be revised to introduce new information. In order to promote the stochastic π-calculus based modeling of biological systems, it is important to use the vast knowledge captured in form of ordinary differential equation based models. These models can not only be extended, but they can also be used for stochastic analysis of the biological systems they represent. The approach is more relevant for biochemical systems where the focus is on measurement and dynamics of species' population. Section 2.6 discusses a method to obtain ODE representation from a graphical description of a biochemical system. Our approach takes inspiration from that method and is suitable for handling such ODEs. To the best of our knowledge, no attempt has been made for automated porting of ordinary differential equation based models to the stochastic π-calculus based models. We demonstrated our approach with two case studies discussed in chapter 4 and 5. The complete implementation of these systems have been given in appendix. The results obtained by the stochastic simulations indicate that our approach is acceptable.

The case study of RKIP on the ERK pathway is a widely discussed case study among researchers from theoretical computer science community and has several references in recent publications [CGH04, CVGO05]. The work also gave us an opportunity to come up with a stochastic π-calculus model for RKIP on ERK pathway which has never been performed before. A PEPA model of RKIP-ERK was presented by Calder et al [CGH04], but the paper does not give sufficient mathematical details of the results produced by the model, otherwise, it could have been useful to *quantitatively* compare the BioSPI model with the PEPA model.

The present work doesn't require the use of mobility feature of π-calculus. It is the first step in developing a process algebra model with more details. The primary goal of this approach is to port the existing ODE models to a process-algebraic environment. Those models can be further extended with more biological information about the system. An example can be found in the BioSPI model of bacterial transcription [Kut05], where biologically observed phenomena were modeled in form of communication to create a π-calculus model.

Our approach is weak in handling of differential equations having complicated terms. For example, we look at a small sub-system of Cell cycle regulation [JJT01] proposed by Tyson and Novak defined by the following differential equations :

$$\frac{d[CycB]}{dt} = k_1 - (k_2' + k_2"[Cdh1])[CycB] \tag{6.1}$$

$$\frac{d[Cdh1]}{dt} = \frac{(k_3' + k_3"A)(1 - [Cdh1])}{J_3 + 1 - [Cdh1]} - \frac{k_4 m[CycB][Cdh1]}{J_4 + [Cdh1]} \tag{6.2}$$

The above ODEs have more complicated forms than the ODEs used in our case-studies. These ODEs involve normalized terms, Michaelis constants as J's in the system and representations like (1 - [Cdh1]), which simply means concentration of non-active Cdh1, if the total amount of Cdh1 present in the system is 1 unit. We cannot simply apply the method discussed in previous chapters for cases like these. The system can still be captured in the form of ODE with progressive and decay terms, but the handling of variables in this case requires improved mathematical approaches. Handling of Michaelis-Menten constant during conversion of deterministic to stochastic modeling is still an open area for discussion [RA03, MSA05, Gou05], and our proposed methodology should be extended to interpret more complicated terms in differential equations in a better way.

Our proposed method for porting ODE based models to stochastic π-calculus models is partially automated. We implement original models described in form of ODEs on the Matlab platform for deterministic simulation. The Matlab code for stochastic simulation of reaction-centric model is generic, only the input matrix (having entries $0, +1, -1$) needs to be adjusted according to the system. The formulation of reactant-centric model in form of chemical reactions is automated. This process of automation can be extended for automatic generation of stochastic π-calculus code. A math routine can be implemented to convert deterministic reaction rates to corresponding base rates for the channels.

Another important problem arises, where the stochastic version might not be a suitable modeling strategy for a system. This question has to do with the choice of model: deterministic or stochastic. If some species exist in small quantities (eg. 4,7,10 etc.), stochastic modeling is the only way to model such systems. This is also true when the system has switching behaviour. A deterministic model (ODE based kinetics) should be used when we have large numbers of entities (like 30000,40000 etc.) in the system. If our system lies in one of the above groups, the usual methods will work, otherwise, if we are in a situation where some numbers are in magnitude like 1,2,4..and some in magnitude of 20000,30000 which is very likely in a single model in real life modeling, we require a '*hybrid approach*'. Hybrid approach requires some of the parts of a model to be modeled using deterministic approach and some with stochastic approach. Extending our work to accommodate hybrid approaches of modeling, and combining it with the expressive power of stochastic π-calculus syntax can prove to be a very useful approach.

We need to extend BioSPI's mathematical engine. The present engine supports only the Gillespie algorithm for mathematical simulation of a system. A recent version of BioSPI (version 3) has a support for writing BioAmbient programs. BioAmbients is a modified version of Ambient π-calculus proposed by Cardelli and Gordon [AR04]. BioAmbients can provide abstraction for compartments in a cell. Compartments introduce a notion of *location*. Many entities in a biological system may be *within* or *outside* a given compartment. One of the features of the π-calculus to capture the internal structure of a biological entity can be realised with this approach. But the simulation still lacks the notion of 3-dimensional space. The first step in this direction would be to handle space as a lattice of ambients, and diffusion of molecules as movement across them.

The lack of graphical representation for π-calculus models is a major disadvantage of this approach. The whole approach of using the stochastic π-calculus as a tool for modeling can become very appealing if we can provide graphical visualization for a model. Phillips and Cardelli [PC04] proposed a graphical representation of their version of stochastic π-calculus, a similar methodology can be developed for our version of stochastic π-calculus also.

A model checker can be developed for models developed in stochastic π-calculus. Stochastic model checking can be a useful tool for quantitative analysis of queries like, if the population of a certain species reaches a particular point, it will remain at that level thereafter.

Bibliography

[AA98] McAdams HH. Arkin A, Ross J. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics.*, 149(4):1633–48, 1998.

[AACH03] David P. Bullivant Autumn A. Cuellar, Poul F. Nielsen and Peter J. Hunter-Nov. Cellml 1.1 for the definition and exchange of biological models. *Conf. Proc. 2003 IFAC Symposium on Modelling and Control in Biomedical Systems*, pages 451–456, 2003.

[AR04] William Silverman Luca Cardelli Ehud Shapiro Aviv Regev, Ekaterina M. Panina. Bioambients: An abstraction for biological compartments. *Theoretical Computer Science*, 325(1):141–167, September 2004.

[Car] Luca Cardelli. Abstract machines of systems biology. *Springer, To appear.*

[Car04a] Luca Cardelli. Bioware languages. In Karen Sprck Jones Andrew Herbert, editor, *Computer Systems: Theory, Technology, and Applications - A Tribute to Roger Needham, Monographs in Computer Science*, pages 59–65. Springer, 2004.

[Car04b] Luca Cardelli. Process calculi and biology. Position Paper for IST FET, 2004.

[Car05] Luca Cardelli. Biological systems as complex systems. Position Paper for IST FET Complex Systems, 2005.

[CGH04] M. Calder, S. Gilmore, and J. Hillston. Modelling the influence of rkip on the erk signalling pathway using the stochastic process algebra pepa. In *Proceedings of Bio-CONCUR 2004*, 2004.

[CGH05] M. Calder, S. Gilmore, and J. Hillston. Automatically deriving odes from process algebra models of signalling pathways. In *Proceedings of CMSB 2005*, 2005.

[cop05] *Copasi Manual*, 2005.

[CSK+03] Kwang-Hyun Cho, Sung-Young Shin, Hyun Woo Kim, Olaf Wolkenhauer, Brian McFerran, and Walter Kolch. Mathematical modeling of the influence of rkip on the erk signaling pathway. In *CMSB '03: Proceedings of the First International Workshop on Computational Methods in Systems Biology*, pages 127–141, London, UK, 2003. Springer-Verlag.

[CVGO05] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of signalling pathways using the prism model checker. In *Proceedings of CMSB 2005*, 2005.

[Dav03] McClay D. R. Hood L. Davidson, E. H. Regulatory gene networks and the properties of the developmental process. *Proc. Natl. Acad. Sci. U. S. A.*, 100:1475–1480, 2003.

[Feh68] E. Fehlberg. Classical fifth, sixth, seventh, and eighth order runge-kutta formulas with stepsize control. Technical Report TR R–287, NASA, 1968.

[FK03] Tanimura N. Morohashi M. Funahashi, A. and H. Kitano. Celldesigner: a process diagram editor for gene-regulatory and biochemical networks. *BIOSILICO*, 1:159–162, 2003.

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

[GH94] S. Gilmore and J. Hillston. The pepa workbench: A tool to support a process algebra-based approach to performance modelling. In *Lecture Notes in Computer Science*, volume 794, pages 353–368. Springer-Verlag, May 1994.

[GIC+01] Bader G.D., Donaldson I., Wolting C., Ouellette B.F., Pawson T., and Hogue C.W. Bind–the biomolecular interaction network database. *Nucleic Acids Res.*, 29:242–245, 2001.

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

[Gil01] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.*, 115:1716–1733, July 2001.

[GK01] Carel Van Gend and Ursula Kummer. Stode - automatic stochastic simulation of systems described by differential equations. In Morohashi Kitano

Yi, Hucka, editor, *Proceedings of the 2nd International Conference on Systems Biology*, pages 326–333. Omnipress, Madison, USA, 2001.

[Gou05] John Goutsias. Quasiequilibrium approximation of fast reaction kinetics in stochastic biochemical systems. *The Journal of Chemical Physics*, 122, 2005.

[GP04] D.T. Gillespie and L.R. Petzold. Improved leap-size selection for accelerated stochastic simulation. *J. Chem. Phys.*, 119:8229–8234, 2004.

[JJT01] Bela Novak John J. Tyson. Regulation of the eukaryotic cell cycle: Molecular antagonism, hysteresis, and irreversible transitions. *Journal of Theoretical Biology*, 210:249–263, 2001.

[Kar77] W J Karplus. The place of systems ecology models in the spectrum of mathematical models. In G.S.Innis, editor, *New directions in the Analysis of Ecological Systems.Part-2*. 1977.

[Kau93] S.A. Kauffman. *The origins of order: Self-organization and selection in evolution.* Oxford University Press, New York., 1993.

[Kit02a] Hiroaki Kitano. Computational system biology. *Nature*, 420:206–210, November 2002.

[Kit02b] Hiroaki Kitano. System biology: A brief overview. *Science*, 295:1662–1664, March 2002.

[KN04] Celine Kuttler and Joachim Niehren. Gene regulation in the pi calculus: simulating cooperativity at the lambda switch. BioConcur 2004, 2004.

[Koh99] Kurt W. Kohn. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molec. Biol. Cell*, 10:2703–2734, 1999.

[KT03] Y. Sadamoto S. Ohta A. Shiozawa F. Miyoshi Y. Naito Y. Nakayama M. Tomita K. Takahashi, N. Ishikawa. E-cell 2: Multi-platform e-cell simulation system. *Bioinformatics*, 19:13:1727–1729, 2003.

[Kut05] Celine Kuttler. Bacterial transcription in the pi calculus. 3rd International Workshop on Computational Methods in Systems Biology, April 2005.

[Lot25] A. J. Lotka. *Elements of physical biology.* Williams & Wilkins Co., 1925.

[LYY03] A. Hoonlor L. You and J. Yin. Modeling biological systems using dynetica v a simulator of dynamic networks. *Bioinformatics*, 19:435–436, 2003.

[MF98] C. J. Morton-Firth. *Stochastic Simulation of Cell signaling Pathways.* PhD thesis, University of Cambridge, Cambridge, UK., 1998.

[Mil99] Robin Milner. *Communicating and Mobile Systems:The p-Calculus.* Number ISBN 0521658691. Cambridge University Press, 1999.

[MKP01] G. Norman M. Kwiatkowska and D. Parker. Prism: Probabilistic symbolic model checker. Technical Report 760/2001, University of Dortmund, 2001. In Proc. PAPM/PROBMIV'01 Tools Session.

[ML98] William Loomis Michael Laub. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Molecular Biology of the Cell*, 9:3521–3532, 1998.

[MSA05] Sergey Plyasunov Michael Samoilov and Adam P. Arkin. Stochastic amplification and signaling in enzymatic futile cycles through noise-induced bistability with oscillations. *PNAS*, 102:2310–2315, 2005.

[NKH01] I.R. Cohen N. Kam and D. Harel. The immune system as a reactive system: Modeling t cell activation with statecharts. In *Proc. Visual Languages and Formal Methods (VLFM'01), part of IEEE Symp. on Human-Centric Computing (HCC'01)*, 2001.

[OGS+99] H. Ogata, S. Goto, K. Sato, W. Fujibuchi, H. Bono, and M. Kanehisa. Kegg: Kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, 27:29–34, 1999.

[PC04] Andrew Phillips and Luca Cardelli. A correct abstract machine for the stochastic pi-calculus. In *Bioconcur'04*. ENTCS, August 2004.

[PJEG98] Jean Peccoud Peter J. E. Goss. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proc Natl Acad Sci U S A*, 95(12):67506755, 1998.

[pria] Prism case study - cell cycle. url - http://www.cs.bham.ac.uk/~dxp/prism/casestudies/cyclin.php, retrieved on 8th october,2005.

[prib] Prism case study - molecular reactions. url - http://www.cs.bham.ac.uk/~dxp/prism/casestudies/molecules.php, retrieved on 8th october,2005.

[Pri95] Corrado Priami. Stochastic pi-calculus. *The Computer Journal*, 38:578–589, 1995.

[PRSS01] Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Inf. Process. Lett.*, 80(1):25–31, 2001.

[RA03] Christopher V. Rao and Adam P. Arkin. Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the gillespie algorithm. *The Journal of Chemical Physics*, 118:4999–5010, 2003.

[Reg02] Aviv Regev. *Computational System Biology: A Calculus for Biomolecular Knowledge*. PhD thesis, Tel Aviv University, 2002.

[RSS01] Aviv Regev, William Silverman, and Ehud Y. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[Sha87] E. Shapiro. *Concurrent Prolog*, volume 1. MIT Press, 1987.

[Voi00] Eberhard O. Voit. *Computational Analysis of Biochemical Systems*. Cambridge University Press, 2000.

[Vol26] V. Volterra. *Variazioni e fluttuazioni del numero d'individui in specie animali conviventi*. Mem. R. Accad. Naz. dei Lincei., 1926.

[W.Sa] A.Houri E.Shapiro W.Silverman, M.Hirsch. Logix user manual for system 2.0. logix system distribution documents. Technical report.

[W.Sb] A.Houri E.Shapiro W.Silverman, M.Hirsch. Suppliment to user manual for system 2.0. logix system distribution documents. Technical report.

Appendix A

Computational Models

A.1 BioSPI implementation of Case-study I

File : rates.cp

```
Rate_1 => 0.0072.
Rate_2 => 0.0315 .
Rate_3 => 0.0087 .
Rate_4 => 0.0072 .
Rate_5 =>  0.87 .
Rate_6 =>  0.0087 .
Rate_7 =>  0.0087 .
Rate_8 =>  0.00245 .
Rate_9 =>  0.0072 .
Rate_10 => 0.0103 .
Rate_11 => 0.0103 .
Rate_12 => 0.00245 .
Rate_13 => 0.0315 .
Rate_14 => 0.0315 .
Rate_15 => 0.0075 .
Rate_16 => 0.0132 .
Rate_17 => 0.0315 .
Rate_18 => 0.00122 .
Rate_19 => 0.0152 .
Rate_20 => 0.0075 .
Rate_21 => 0.071 .
Rate_22 => 0.0132.
Rate_23 => 0.0132.
Rate_24 => 0.071 .
Rate_25 => 0.0075 .
Rate_26 => 0.071 .
Rate_27 => 0.00245 .
Rate_28 => 0.0103 .
Rate_29 => 0.00122 .
Rate_30 => 0.87 .
Rate_31 => 0.0152 .
Rate_32 =>  0.0152 .
Rate_33 => 0.00122 .
```

```
Rate_34 => 0.87 .
```

File : erk_spi.cp

```
-language(spifcp).
-include(rates).

global(
 reaction_1(Rate_1),
 reaction_2(Rate_2),
 reaction_3(Rate_3),
 reaction_4(Rate_4),
 reaction_5(Rate_5),
 reaction_6(Rate_6),
 reaction_7(Rate_7),
 reaction_8(Rate_8),
 reaction_9(Rate_9),
 reaction_10(Rate_10),
 reaction_11(Rate_11),
 reaction_12(Rate_12),
 reaction_13(Rate_13),
 reaction_14(Rate_14),
 reaction_15(Rate_15),
 reaction_16(Rate_16),
 reaction_17(Rate_17),
 reaction_18(Rate_18),
 reaction_19(Rate_19),
 reaction_20(Rate_20),
 reaction_21(Rate_21),
 reaction_22(Rate_22),
 reaction_23(Rate_23),
 reaction_24(Rate_24),
 reaction_25(Rate_25),
 reaction_26(Rate_26),
 reaction_27(Rate_27),
 reaction_28(Rate_28),
 reaction_29(Rate_29),
 reaction_30(Rate_30),
 reaction_31(Rate_31),
 reaction_32(Rate_32),
 reaction_33(Rate_33),
 reaction_34(Rate_34)
).


System(N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11)::= <<

           CREATE_M1(N1) |
       CREATE_M2(N2) |
       CREATE_M3(N3) |
       CREATE_M4(N4) |
       CREATE_M5(N5) |
       CREATE_M6(N6) |
       CREATE_M7(N7) |
       CREATE_M8(N8) |
       CREATE_M9(N9) |
       CREATE_M10(N10) |
       CREATE_M11(N11) |
         Timer(reaction_1) | Timer(reaction_2) |
```

```
          Timer(reaction_4) | Timer(reaction_5) |
          Timer(reaction_8) | Timer(reaction_9) |
          Timer(reaction_12) | Timer(reaction_13) |
          Timer(reaction_14) | Timer(reaction_15) |
          Timer(reaction_17) | Timer(reaction_18) |
          Timer(reaction_20) | Timer(reaction_21) |
          Timer(reaction_24) | Timer(reaction_25) |
          Timer(reaction_26) | Timer(reaction_27) |
          Timer(reaction_29) | Timer(reaction_30) |
          Timer(reaction_33) | Timer(reaction_34) .



  CREATE_M1(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M1 | self .

  CREATE_M2(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M2 | self .

  CREATE_M3(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M3 | self .
  CREATE_M4(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M4 | self .

  CREATE_M5(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M5 | self .

  CREATE_M6(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M6 | self .
  CREATE_M7(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M7 | self .

  CREATE_M8(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M8 | self .

  CREATE_M9(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M9 | self .
  CREATE_M10(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M10 | self .

  CREATE_M11(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | M11 | self

>> .


Timer(channel)::= channel ! [] , Timer .


M3 ::= reaction_1 ? [] , M3 | M1 ;
       reaction_4 ? [] , M3 | M2 ;
       reaction_9 ? [], true ;
       reaction_10 ? [], true ;
       reaction_11 ? [], M3 ;
       reaction_28 ? [], M3 .

M4 ::= reaction_2 ? [], M4 | M1 ;
       reaction_8 ? [], M4 | M3 ;
       reaction_12 ? [], true ;
       reaction_13 ? [], true ;
       reaction_14 ? [], M4 | M5 ;
```

```
       reaction_17 ? [], M4 | M6 ;
       reaction_27 ? [], M4 | M9 .


M1 ::= reaction_3 ? [] , true ;
       reaction_6 ? [], M1 ;
       reaction_7 ? [], M1 .


M2 ::= reaction_3 ! [] , M2 ;
       reaction_6 ! [], true ;
       reaction_7 ! [], M2 | M3 .


M11 ::= reaction_5 ? [], M11 | M2 ;
        reaction_18 ? [], M11 | M6 ;
        reaction_29 ? [], M11 | M10 ;
        reaction_30 ? [], M11 | M10 ;
        reaction_33 ? [], true ;
        reaction_34 ? [], true .

M9 ::= reaction_10 ! [], M9 ;
       reaction_11 ! [], M9 | M4 ;
       reaction_28 ! [], true .

M8 ::= reaction_15 ? [], M8 | M5 ;
       reaction_20 ? [], M7 | M8 ;
       reaction_21 ? [], M7 | M8 ;
       reaction_24 ? [], true ;
       reaction_25 ? [], true ;
       reaction_26 ? [], M8 | M9 .

M5 ::= reaction_16 ? [], true ;
       reaction_22 ? [], M5 ;
       reaction_23 ! [], M5 .


M7 ::= reaction_16 ! [], M7 ;
       reaction_22 ! [], true ;
       reaction_23 ? [], M7 | M8 .

M6 ::= reaction_19 ? [], true ;
       reaction_31 ? [], M6 ;
       reaction_32 ! [], M6 .

M10 ::= reaction_19 ! [], M10 ;
        reaction_31 ! [], true ;
        reaction_32 ? [], M10 | M11 .
```

A.2 BioSPI implementation of Case-study II

File : rates_dicto.cp

```
Rate_1 => 1.4 .
Rate_2 => 0.9 .
Rate_3 => 2.5 .
```

```
Rate_4 => 1.5 .
Rate_5 => 0.6 .
Rate_6 => 0.0132 .
Rate_7 => 120.46 .
Rate_8 => 0.0215 .
Rate_9 => 0.3 .
Rate_10 => 0.0132 .
Rate_11 => 0.7.
Rate_12 => 4.9 .
Rate_13 => 18.0 .
Rate_14 => 0.0249 .
```

File : dict_spi.cp

```
-language(spifcp).
-include(rates_dicto).

global(
 reaction_1(Rate_1),
 reaction_2(Rate_2),
 reaction_3(Rate_3),
 reaction_4(Rate_4),
 reaction_5(Rate_5),
 reaction_6(Rate_6),
 reaction_7(Rate_7),
 reaction_8(Rate_8),
 reaction_9(Rate_9),
 reaction_10(Rate_10),
 reaction_11(Rate_11),
 reaction_12(Rate_12),
 reaction_13(Rate_13),
 reaction_14(Rate_14)
).


System(N1,N2,N3,N4,N5,N6,N7,N8)::= <<

           CREATE_ACA(N1) |
       CREATE_PKA(N2) |
       CREATE_ERK2(N3) |
       CREATE_REGA(N4) |
       CREATE_CAMPI(N5) |
       CREATE_CAMPE(N6) |
       CREATE_CAR1(N7) | CREATE_Dummy(N8) |
       Timer(reaction_1)| Timer(reaction_2)|
       Timer(reaction_3)| Timer(reaction_4)|
       Timer(reaction_5)| Timer(reaction_7)|
       Timer(reaction_9)| Timer(reaction_11) |
       Timer(reaction_12) | Timer(reaction_13) .


  CREATE_ACA(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | ACA | self .

  CREATE_PKA(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | PKA | self .

  CREATE_ERK2(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | ERK2 | self .
```

```
  CREATE_REGA(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | REGA | self .

  CREATE_CAMPI(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | CAMPI | self .

  CREATE_CAMPE(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | CAMPE | self .

  CREATE_CAR1(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | CAR1 | self .

  CREATE_Dummy(C)::= {C =< 0} , true ;
      {C > 0} , {C--} | Dummy | self

>> .


Timer(channel)::= channel ! [] , Timer .


ACA ::= reaction_2 ? [] , true;
reaction_9 ? [] , ACA | CAMPI;
        reaction_11 ? [], ACA | CAMPE .

ERK2 ::= reaction_1 ? [] , ERK2 | ACA ;
 reaction_6 ? [] , true;
 reaction_8 ! [] , ERK2.


CAMPI ::= reaction_3 ? [] , CAMPI | PKA ;
          reaction_10 ! [] , true.


PKA ::= reaction_4 ? [] , true ;
        reaction_6 ! [] , PKA ;
        reaction_14 ! [] , PKA .

CAR1 ::= reaction_5 ? [] , CAR1 | ERK2 ;
         reaction_14 ? [] , true.


REGA ::= reaction_8 ? [], true;
         reaction_10 ? [], REGA.

CAMPE ::= reaction_12 ? [] , true;
          reaction_13 ? [] , CAMPE | CAR1 .


Dummy ::= reaction_7 ? [] , Dummy | REGA .
```

A.3 Implementation of RKIP-ERK pathway using the Gillepie algorithm

File : rkip_reaction.mat

```
% This file contains the description of Chemical reactions
that occur in RKIP pathway
% m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11
-1 -1 +1 0 0 0 0 0 0 0 0
+1 +1 -1 0 0 0 0 0 0 0 0
0 0 -1 +1 0 0 0 0 -1 0 0
0 0 +1 -1 0 0 0 0 +1 0 0
+1 0 0 -1 +1 +1 0 0 0 0 0
0 0 0 0 +1 0 +1 -1 0 0 0
0 0 0 0 -1 0 -1 +1 0 0 0
0 0 0 0 0 0 +1 -1 +1 0 0
0 0 0 0 +1 0 0 0 0 +1 -1
0 0 0 0 0 -1 0 0 0 -1 +1
0 +1 0 0 0 0 0 0 0 +1 -1
```

File : rkip_initialsetup.m

```
% Settings for the stochastic analysis of RKIP-pathway

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulation Parameter Setting space
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all,clc,close all

% volume of the compartment
V = 1.0e-22;

% Deterministic constants
k = [0.53 0.0072 0.625 0.00245 0.0315 0.8 0.0075 0.071 0.92 0.00122 0.87];

% Starting population in Moles/Liter
start_population = [2.5 2.5 0 0 0 0 2.5 0 2.5 3 0];

% Simulation ends in these many seconds
tf = 100;

R = load('rkip_reaction','-ascii');
[m,n] = size(R);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Processing starts here.................
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% reset random number generator
rand('state',sum(100*clock))

%NAV = Avogadro's number times volume;
NAV = V * 6.02214199e23;

% Conversion of population into molecules
n0 = round(start_population * NAV);

% Empty vector for 'm' reactions
c = zeros(1,m);

% Compute the stochastic rate constants
for i=1:m
```

```
    % pick up a reaction
    reaction = R(i,:);

    % Extract the number of entities on LHS of reaction, -ve entries are on
    % LHS side
    no_of_reactants = 0;
    for j = 1:n
        if R(1,j) < 0
            no_of_reactants = no_of_reactants + 1;
        end
    end

    if no_of_reactants == 0
        print('Error in Reaction Matrix specification');
    end

    c(1,i) = (k(1,i) * (no_of_reactants - 1)) / (NAV^(no_of_reactants - 1));
end


t = 0;
tt = 0;
n = n0;
nn = n0;

aa = rkip_propensity(n0,R,c);

while t < tf

    a = rkip_propensity(n,R,c);
    astr = sum(a);

    if ~astr
        t = tf;
    else

        % Compute the next reaction time
        tau = exprnd(inv(astr));

        % Find the next reaction to occur
        mu = min( find(cumsum(a) > astr * rand) );

        % update time
        t = t + tau;

        % change the species population
        n = n + R(mu,:);

    end

    tt = [tt ; t];
    nn = [nn ; n];
    aa = [aa ; a];

end

tt(end) = tf;
nn(end) = nn(end -1);
aa(end) = aa(end -1);
```

File : rkip_propensity.m

```
% Propensity function for handling the population of species
% and also calculating the 'a' term of the Gillespie algorithm

% population_matrix -> Population of Species
% reaction_matrix -> m x n matrix of 'm' reactions and 'n' species
% c_matrix -> Stochastic rate constants for 'm' rections
% returns 'a' vector for 'm' recations

function a = rkip_propensity(population_matrix,reaction_matrix,c_matrix)

% m = number of reactions
% n = number of species
[m,n] = size(reaction_matrix);

h = zeros(1,m);

for i=1:m

    % Pick up a reaction
    reaction = reaction_matrix(i,:);

    % We want to keep only those terms which take part in reaction
    % So, if we multiply the population matrix with the reaction
    % matrix, only the participating indices will be non-zero.
    temp_population = population_matrix .* reaction;

    % Pick up those terms which are -ve as those are the species
    % which are consumed and contribute towards calculation of 'h'.
    % Multiply them together , taking their magnitude only

    prod = 1;
    for j = 1:n
        if temp_population(1,j) < 0
            prod = prod * temp_population(1,j) * (-1);
        end
    end

    if prod > 1
        h(1,i) = prod;
    else
        h(1,i) = 0;
    end

end

% Compute 'a'
a = c_matrix.* h;
```

Appendix B

Figures from The Case study of RKIP on ERK Pathway

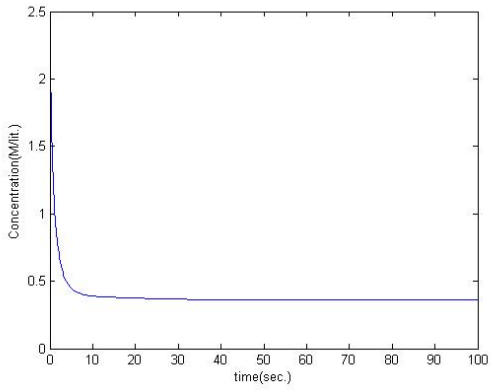

Figure B.1: Raf-1* (Deterministic simulation)

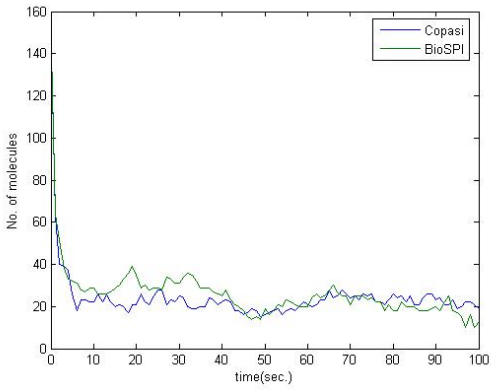

Figure B.2: Raf-1* (Stochastic simulation)

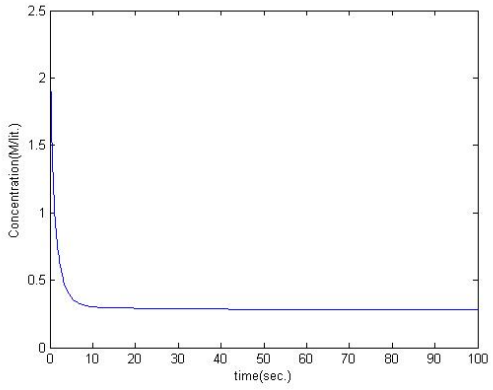

Figure B.3: RKIP (Deterministic simulation)

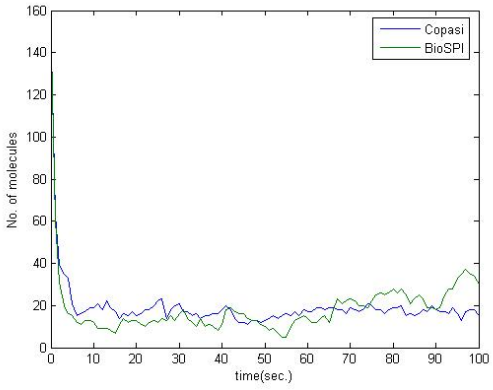

Figure B.4: RKIP (Stochastic simulation)

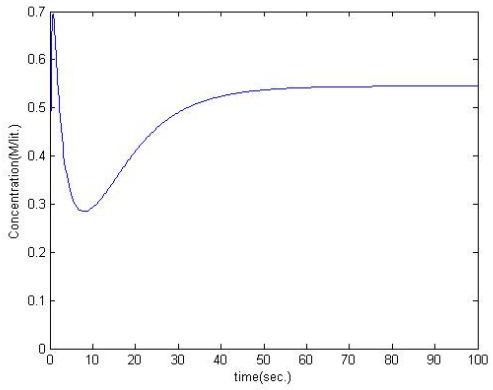

Figure B.5: Raf-1*/RKIP (Deterministic simulation)

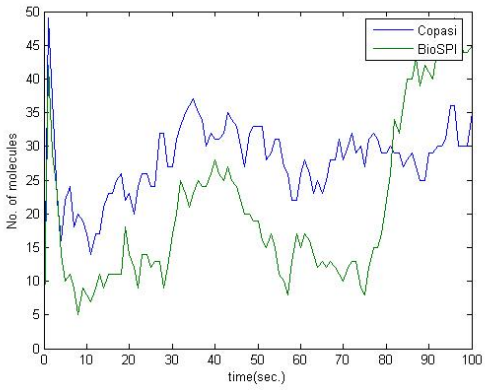

Figure B.6: Raf-1*/RKIP (Stochastic simulation)

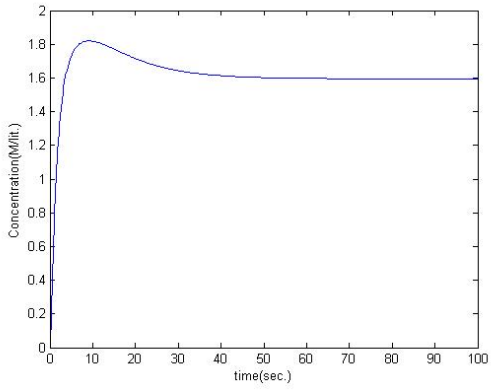

Figure B.7: Raf-1*/RKIP/ERK-PP (Deterministic simulation)

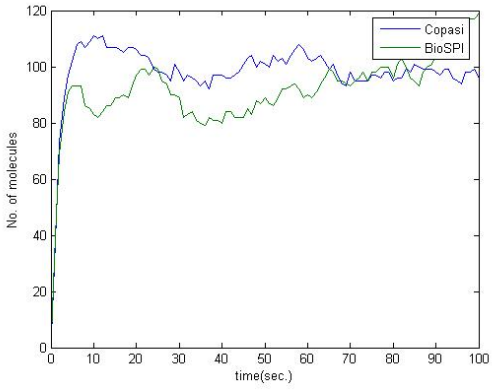

Figure B.8: Raf-1*/RKIP/ERK-PP (Stochastic simulation)

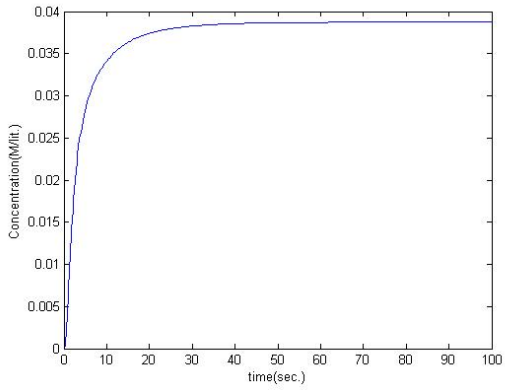

Figure B.9: ERK (Deterministic simulation)

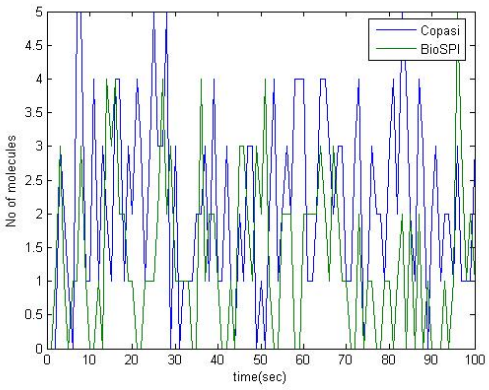

Figure B.10: ERK (Stochastic simulation)

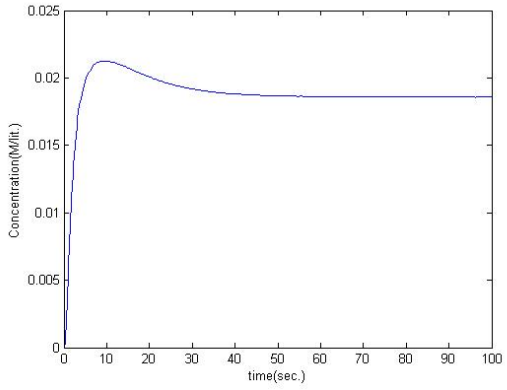

Figure B.11: RKIP-P (Deterministic simulation)

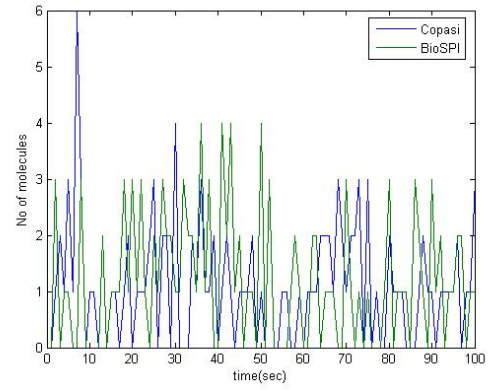

Figure B.12: RKIP-P (Stochastic simulation)

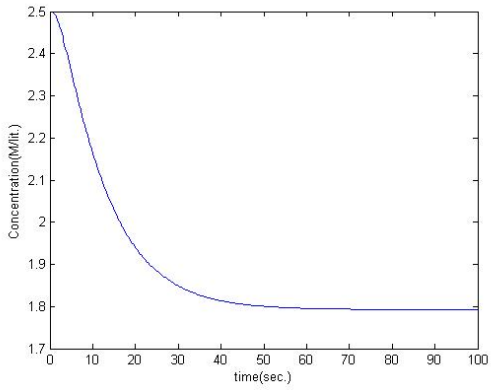

Figure B.13: MEK-PP (Deterministic simulation)

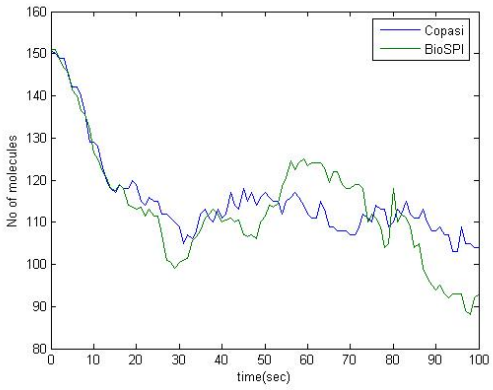

Figure B.14: MEK-PP (Stochastic simulation)

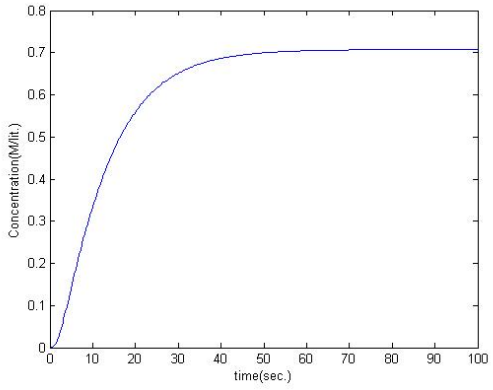

Figure B.15: MEK-PP/ERK (Deterministic simulation)

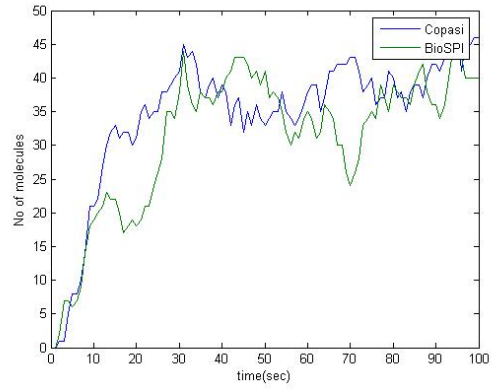

Figure B.16: MEK-PP/ERK (Stochastic simulation)

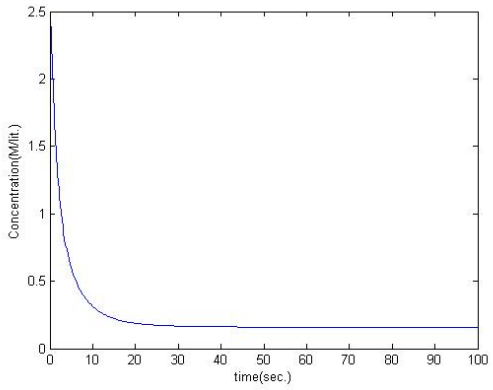

Figure B.17: ERK-PP (Deterministic simulation)

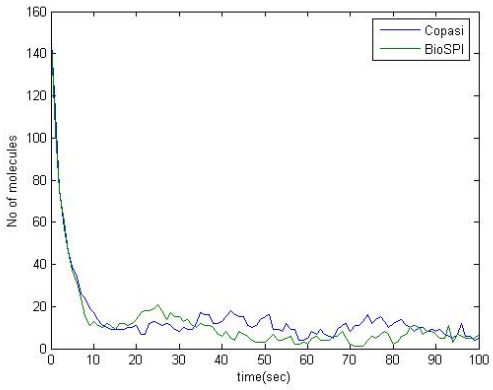

Figure B.18: ERK-PP (Stochastic simulation)

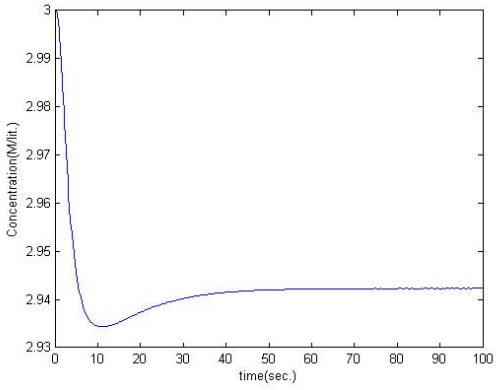

Figure B.19: RP (Deterministic simulation)

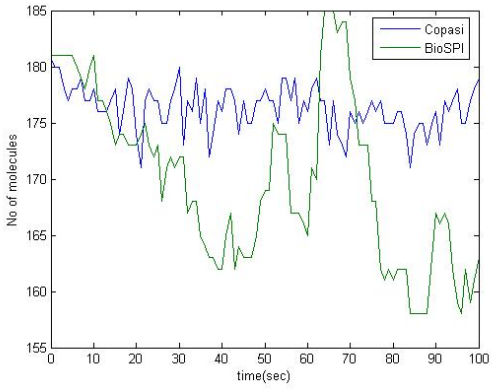

Figure B.20: RP (Stochastic simulation)

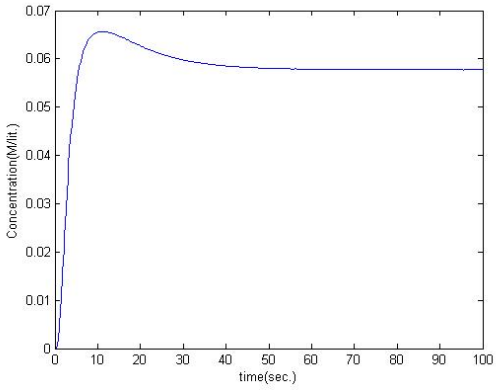

Figure B.21: RKIP-P/RP (Deterministic simulation)

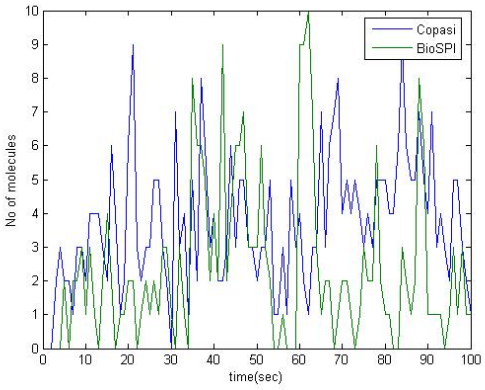

Figure B.22: RKIP-P/RP (Stochastic simulation)

Figure B.23: Screenshot of simulation of RKIP-ERK system in STODE

Figure B.24: Screenshot of simulation of RKIP-ERK in COPASI

Appendix C

Figures from The Case-study of The Dictyostelium Model

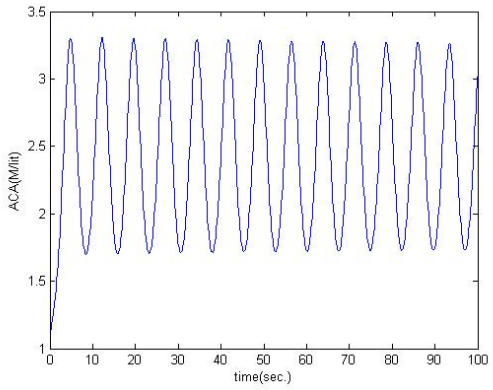

Figure C.1: ACA (Deterministic simulation)

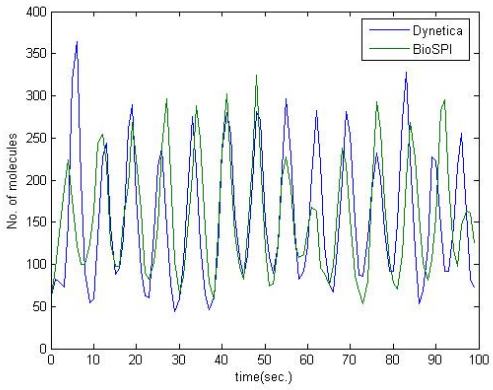

Figure C.2: ACA (Stochastic simulation)

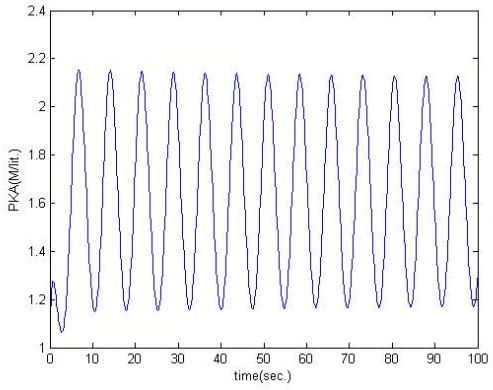

Figure C.3: PKA (Deterministic simulation)

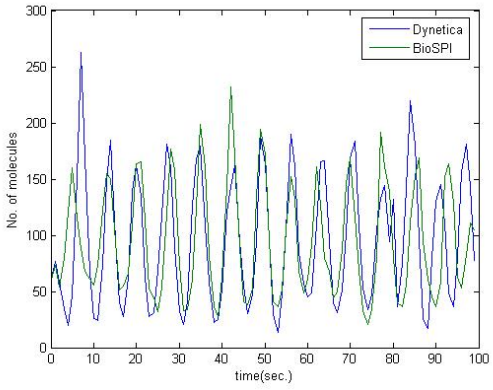

Figure C.4: PKA (Stochastic simulation)

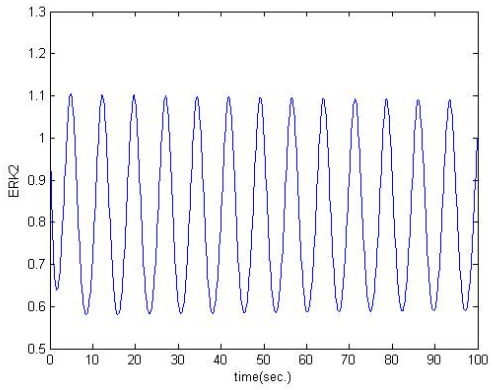

Figure C.5: ERK2 (Deterministic simulation)

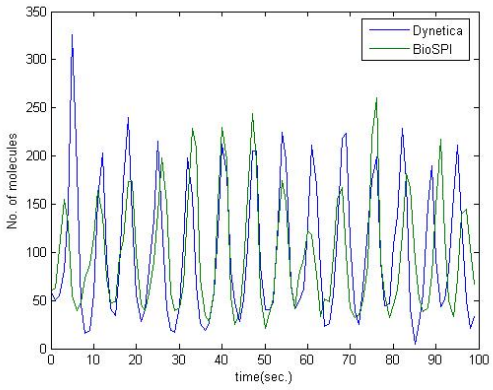

Figure C.6: ERK2 (Stochastic simulation)

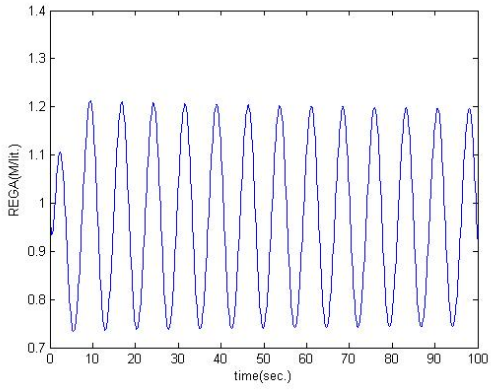

Figure C.7: REGA (Deterministic simulation)

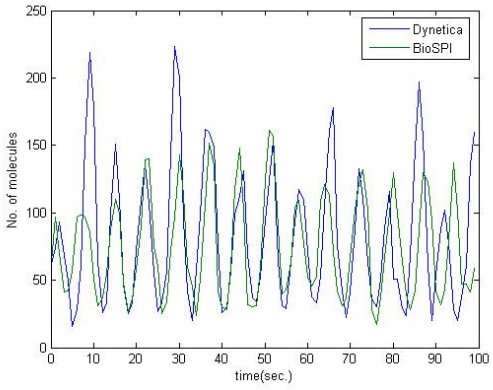

Figure C.8: REGA (Stochastic simulation)

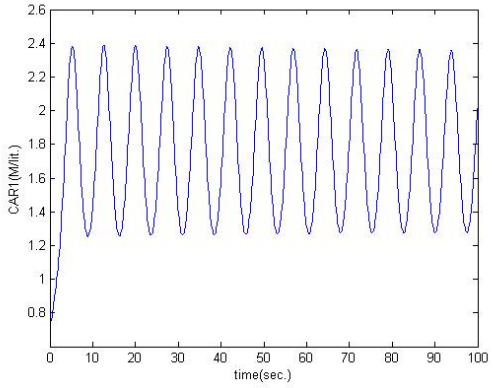

Figure C.9: CAR1 (Deterministic simulation)

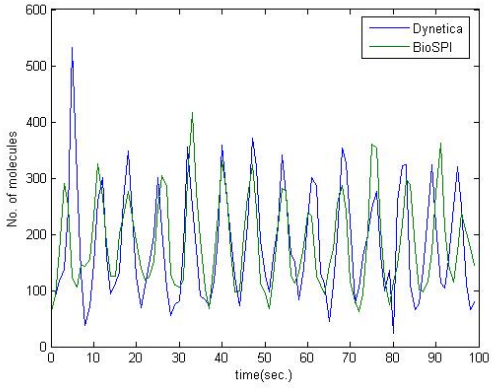

Figure C.10: CAR1 (Stochastic simulation)

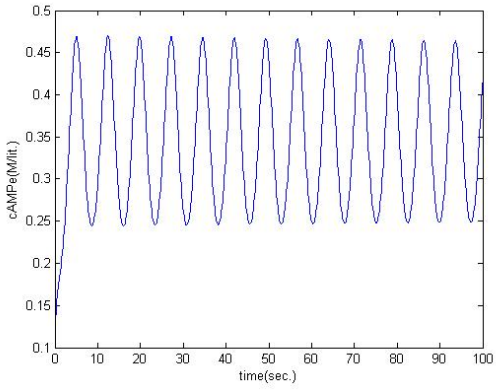

Figure C.11: cAMPe (Deterministic simulation)

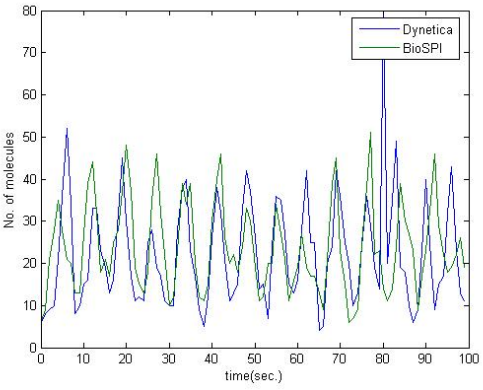

Figure C.12: cAMPe (Stochastic simulation)

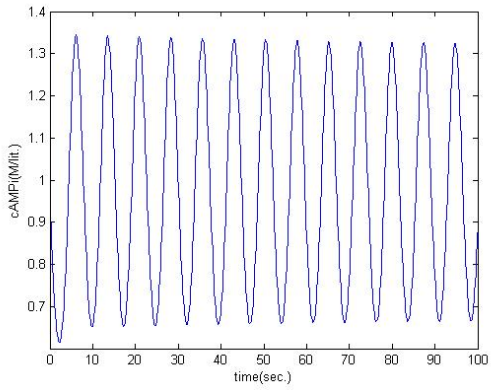

Figure C.13: cAMPi (Deterministic simulation)

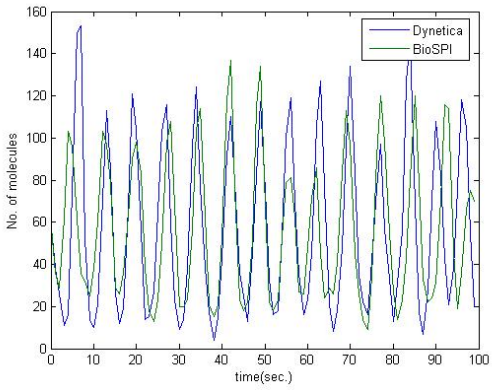

Figure C.14: cAMPi (Stochastic simulation)

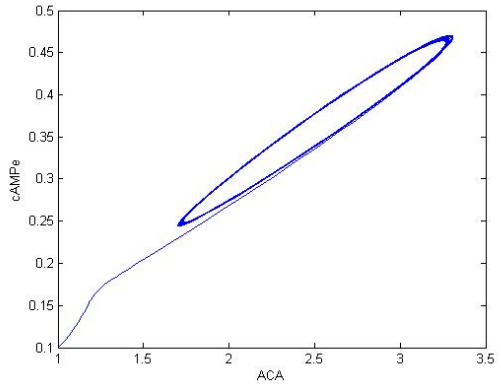

Figure C.15: ACA vs. cAMPe (Deterministic simulation)

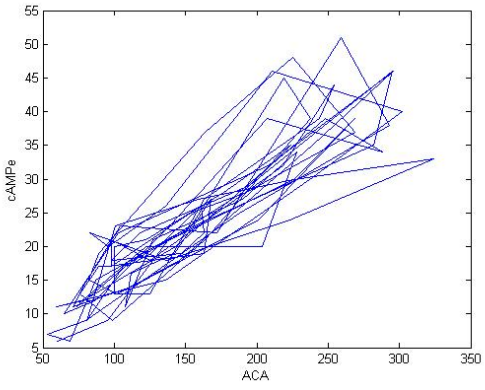

Figure C.16: ACA vs. cAMPe (Stochastic simulation)

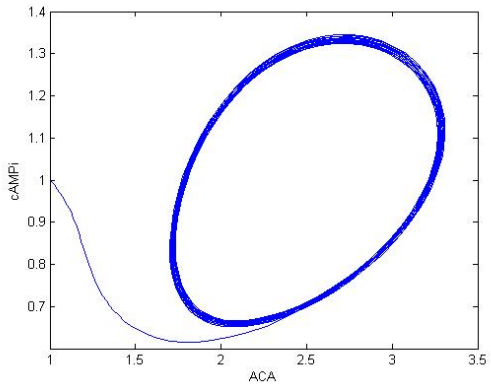

Figure C.17: ACA vs. cAMPi(Deterministic simulation)

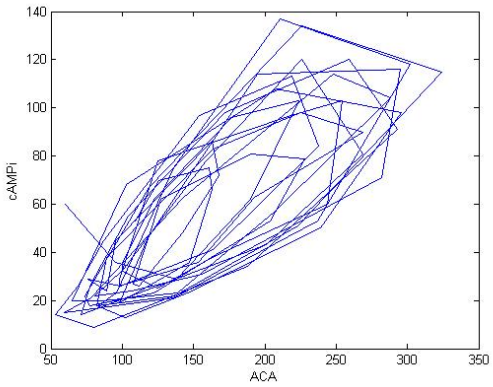

Figure C.18: ACA vs. cAMPi(Stochastic simulation)

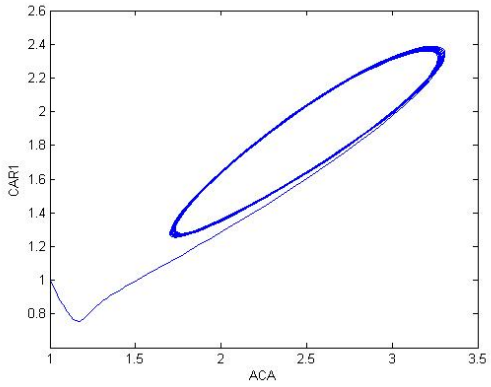

Figure C.19: ACA vs. CAR1(Deterministic simulation)

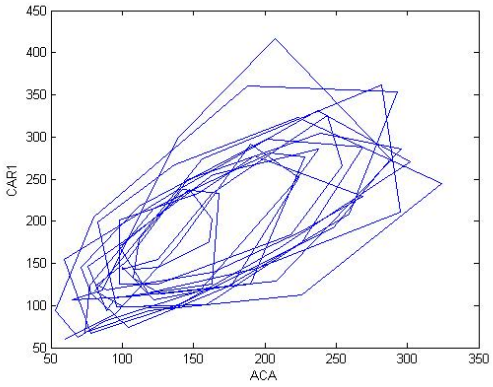

Figure C.20: ACA vs. CAR1(Stochastic simulation)

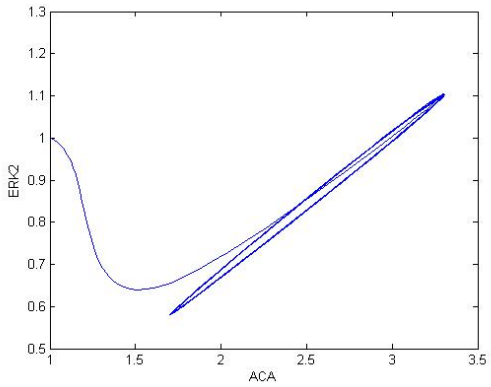

Figure C.21: ACA vs. ERK2(Deterministic simulation)

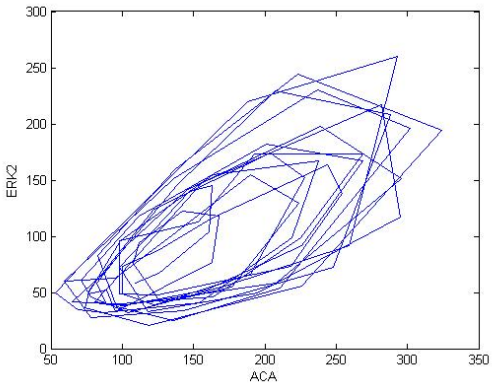

Figure C.22: ACA vs. ERK2(Stochastic simulation)

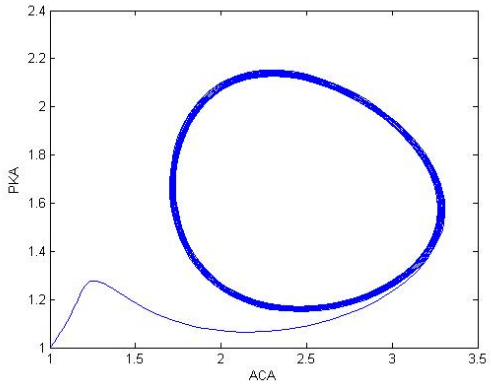

Figure C.23: ACA vs. PKA(Deterministic simulation)

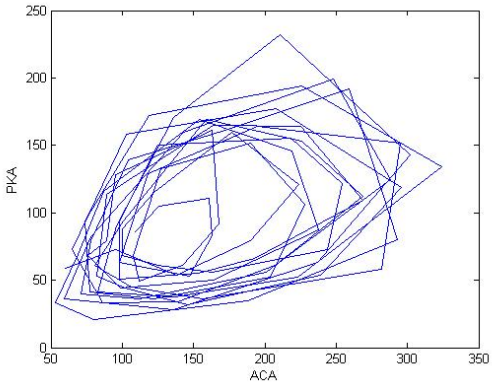

Figure C.24: ACA vs. PKA(Stochastic simulation)

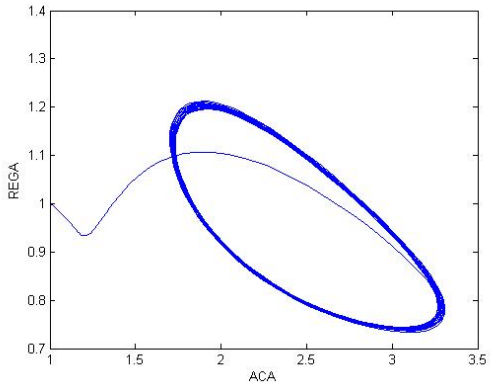

Figure C.25: ACA vs. REGA(Deterministic simulation)

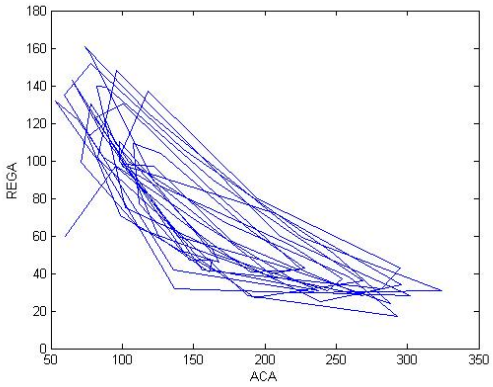

Figure C.26: ACA vs. REGA(Stochastic simulation)