

UNIVERSITÀ DEGLI STUDI DI TRENTO
Facoltà di Scienze Matematiche, Fisiche e Naturali



UNIVERSITY OF TRENTO - Italy

Corso di Laurea Specialistica in Informatica
within European Master in Informatics

Final Thesis

**Stochastic process algebras
as design and analysis framework
for Synthetic Biology modelling.**

Relatore/1st Reader:

Prof. Corrado Priami
University of Trento

Laureando/Student:

Luca Gerosa

Controrelatore/ 2nd Reader:

Prof. Jane Hillston
University of Edinburgh

Anno Accademico 2006 - 2007

Abstract

This dissertation investigates the suitability of the formalism of stochastic process algebras as framework for modelling biological systems in the context of *Synthetic Biology* discipline. *Synthetic Biology* is an emergent field of bio-engineering research with the goal of re-design functional aspects of living organisms by embedding *synthetic gene networks* in the host cells. These gene networks are constructed using composable DNA regions that are present in an open and public genetic component library, called the *Registry of Standard Biological Parts*. The development of frameworks based on mathematical and computational abstract modelling is considered to be of fundamental importance to support the design and analysis construction phases of such synthetic biological systems.

To demonstrate the applicability of the process algebra approach, we firstly develop a formal graphical notation able to represent Synthetic Biology unicellular systems and then we define an automatic conversion to stochastic process algebra models written in the *PEPA* language (Performance Evaluation Process Algebra). The models are constructed following the ‘cell-as-computation’ paradigm and a mechanistic approach, meaning that we represent individual components and processes as abstraction of real biological mechanisms. The use of process algebras brings benefits such as support for compositionality and abstraction in model construction. The PEPA description of systems specifies an underlying mathematical model in the form of *Continuous Time Markov Chain* stochastic processes, with a discrete representation of populations and a probabilistic specification of time aspects. We discuss the major physical, biological and mathematical assumptions that regard the application of such mathematical models.

We show, using pattern examples and a real case study, that the approach allows the representation and analysis of several fundamental biological mechanisms involved in *metabolic, signalling pathways, gene regulatory, and membrane* networks. Analysis is carried out using stochastic simulations based on the *Gillespie’s Algorithm* and probabilistic model checking. We present the derivation and analysis steps using pseudo-code definitions as guidelines for a future implementation of the modelling workbench. Our work is presented as a first tutorial definition of a structured mapping between process algebras and Synthetic Biology systems. We believe that our work may help to attract the attention of research groups working on process algebra formalism to the fertile ground represented by the Synthetic Biology discipline.

Acknowledgements

Infinite thanks to my supervisor in Edinburgh, Prof. *Jane Hillston*, for the helpful discussions and for having encouraged me in choosing a topic I believed in. I would like also to thank Prof. *Corrado Priami* for his support about the dissertation aspects in Trento. My sincere thanks to the European Master in Informatics staff of the University of Trento and of the University of Edinburgh for having help me during this great experience. Thanks for having been part of these amazing two years to, in completely random order, the 109 Dalkeith Road Dream Team in Edinburgh (*Andrea, Giulia* and *Luciano*), the Edinburgh Friends (*Ambar, Hassan, Simon, Marco* and all the others), the awesome Trento University Crew (a mention to *Pise* and *Bishops* for the support), my Lizzana's Warriors crazy friends, the Edinburgh iGEM Team 2007 and *Elisa*. A big hug to my family, for having been always with me when I needed them the most.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(Luca Gerosa)

*“If the doors of perception were cleansed,
every thing would appear to man as it is:
infinite”*

The Marriage of Heaven and Hell
- William Blake -

Table of Contents

1	Introduction	1
1.1	Motivations	3
1.2	Objectives and methods	4
1.3	State-of-the-art	5
1.4	Structure of the dissertation	7
2	Background	9
2.1	Synthetic Biology	9
2.2	Stochastic Process Algebras in Systems Biology	13
2.3	PEPA: process algebra for systems modelling	15
2.4	PRISM: probabilistic model checker and simulator	18
2.5	Continuous Time Markov Chains	20
2.6	The Gillespie’s Algorithm	21
3	A graphical notation for Synthetic Biology	23
3.1	Design objectives	23
3.2	Symbols	25
3.3	Composition rules	28
3.4	System’s definition and constraints	31
4	The PEPA model	33
4.1	A quantitative mechanistic approach	33
4.2	Construction of the PEPA model	35
4.2.1	Defining a species	36
4.2.2	Defining a reaction	36
4.2.3	Defining a operon (or gene) expression	37
4.2.4	Defining a reaction input	37
4.2.5	Defining a reaction output	38

4.2.6	Defining a reaction input-output	39
4.2.7	Defining a promoter repression	39
4.2.8	Defining a promoter activation	40
4.2.9	Defining the final components	40
4.2.10	Defining the root component	40
4.3	Mechanistic interpretation of the stochastic process	41
4.3.1	Reactions	42
4.3.2	Operon (or gene) expressions	44
4.3.3	Transcriptional regulation processes	44
4.4	Translation from PEPA to PRISM	45
5	Modelling and analysis in Synthetic Biology	49
5.1	Common patterns in biochemical reactions' networks	50
5.1.1	Proteins' Degradation	50
5.1.2	Irreversible and Reversible Reactions	52
5.1.3	Enzymatic Reactions	55
5.2	Common patterns in gene regulatory networks	58
5.2.1	Gene Expression	59
5.2.2	Gene Activation and Repression	59
5.2.3	Inducible and Repressible Systems	63
5.2.4	Transcription and Translation Processes	64
5.3	Common patterns in membrane and transport networks	69
5.3.1	Passive Transport	73
5.3.2	Vesicular Transport	73
5.3.3	Receptor-Ligand Interactions	77
5.4	Case study	79
5.4.1	Edinburgh iGEM 2006: the Arsenic Biosensor	79
6	Conclusions	85
6.1	Concluding remarks and critical discussion	85
6.2	Future works	88
6.3	Final observations	89
A	Appendix	91
A.0.1	PRISM Code for the Protein's Degradation example	91
A.0.2	PRISM Code for the Reversible Reaction example	91

A.0.3	PRISM Code for the Enzymatic Reaction example	93
A.0.4	PRISM Code for the Gene Expression example	94
A.0.5	PRISM Code for the Gene Repression and Inducible System example	95
A.0.6	PRISM Code for the Gene Activation and Repressible System example	96
A.0.7	PRISM Code for the Transcription and Translation Processes example	98
A.0.8	PRISM Code for the Passive Transport example	100
A.0.9	PRISM Code for the Vescicle Transport example	100
A.0.10	PRISM Code for the Receptor-Ligand Interactions example .	101
A.0.11	PRISM Code for the Arsenic Biosensor system	103

Bibliography		107
---------------------	--	------------

List of Figures

2.1	Screenshot of the Registry of Standard Biological Parts	12
2.2	The abstraction hierarchy for integrated genetic systems	13
2.3	PEPA Structured Operational Semantics	17
2.4	PEPA Cooperation rule as multiplication of participant rates	18
2.5	An example of PRISM module definition	19
3.1	Example of a type definition	25
4.1	Syntactical structure of generated PEPA components	36
4.2	Definition of the function \mathcal{A} used in root component construction	42
5.1	Protein's Degradation representation	51
5.2	Stochastic simulations for the Protein's Degradation pattern	52
5.3	Probabilistic distribution of the Protein's Degradation pattern space	53
5.4	Reversible Reaction representation	53
5.5	Stochastic simulations for the Irreversible and Reversible Reaction pattern	55
5.6	Probabilities of populations in a specific range for the Reversible Reaction example	56
5.7	Enzymatic Reaction representation	57
5.8	Stochastic simulations for the Enzymatic Reaction pattern	58
5.9	Gene Expression representation	60
5.10	Stochastic simulations for the Gene Expression pattern	60
5.11	Probabilistic distribution of the Gene Expression pattern space	61
5.12	Gene Repression and Inducible System representation	62
5.13	Gene Activation and Repressible System representation	63
5.14	Stochastic simulations for the Gene Repression and Inducible System pattern	67

5.15 Stochastic simulations for the Gene Activation and Repressible System pattern	68
5.16 Transcription and Translation Processes representation	71
5.17 Stochastic simulations for the Transcription and Translation pattern	72
5.18 Passive Transport representation	73
5.19 Stochastic simulations for the Passive Transport pattern	74
5.20 Vesicular Transport representation	75
5.21 Stochastic simulations for the Vesicular Transport pattern	75
5.22 Probabilistic distribution of the Vesicular Transport pattern space	76
5.23 Receptors-Ligand Interactions representation	78
5.24 Stochastic simulations for the Receptor-Ligand Interactions pattern	79
5.25 Arsenic Biosensor system representation	80
5.26 Stochastic simulations for the Arsenic Biosensor system model	83

List of Tables

2.1	Process algebra abstraction for Systems Biology	14
2.2	Continuous Stochastic Logic operators	18
3.1	Symbols for reaction specification and their formal definitions	26
3.2	DNA parts symbols and formal definitions	27
3.3	Connections between DNA parts and species symbols and formal definitions	28
3.4	Symbols composition patterns for reactions	29
3.5	Symbols composition patterns for operon expression	30
3.6	Symbols composition patterns for transcriptional regulation effects	30
4.1	Process Algebra abstraction for Systems Biology adapted to PEPA	34
4.2	Rules for the conversion from PEPA to PRISM	47
5.1	Protein's Degradation PEPA model	51
5.2	Reversible Reaction PEPA model	54
5.3	Enzymatic Reaction PEPA model	57
5.4	Gene expression PEPA model	60
5.5	Gene Repression and Inducible System PEPA model	65
5.6	Gene Activation and Repressible System PEPA model	66
5.7	Transcription and Translation Processes PEPA model	70
5.8	Passive Transport PEPA model	74
5.9	Vesicular Transport PEPA model	77
5.10	Receptor-Ligand Interactions PEPA model	78
5.11	Description of the standard parts in the Arsenic Biosensor system	80
5.12	Arsenic Biosensor PEPA model	82

Chapter 1

Introduction

Synthetic Biology is a multidisciplinary field of research that investigates experimental techniques and theoretical methodologies with the goal to re-design functional aspects of biological entities. The approach is being used for two main purposes: to reproduce emergent biological behaviours in order to study their characteristics and to exploit the potential of living organisms as programmable machines in order to perform useful functions [1].

Fundamental biological discoveries over the last decades contributed in the formulation of paradigms explaining how internal living organisms function, such as the *Central Dogma* information flow or the biomolecules structure composition. Combined with the insights given by the genome projects, this knowledge highlighted the essential role of the genetic code in a cell's behaviour [2]. With the more recent development of powerful and widespread techniques in bioengineering for the extraction, sequencing and insertion of DNA fragments, this opened the door to the re-programming of cell's behaviour. Therefore, efforts particularly concentrated on the genetic engineering of cells by the modification and extensions of their DNA information in order to construct new *synthetic gene networks*. Advances in our ability to control the operating principles that govern living organisms is expected to create new methodologies in biological experiments as well as bring great benefits in a variety of fields such as human therapeutics, molecular fabrication of biomaterials, biosensors and cell computation.

In the current *state-of-the-art*, the construction of a *Synthetic Biology* system is still a “unique, expert-driven research problem with uncertain times to completion, cost and probabilities of success” [3]. Much research is therefore directed to implement-

ing foundational technologies to transform design and construction of devices into a practical, reliable and useful engineering discipline. This shift demands an engineering approach at all levels, with use of concepts such as *abstraction*, *standardization* and *decoupling* as fundamental paradigms. In this context, the *Library of Standard Biological Parts* is a recent attempt to construct a collection of functional and composable DNA parts that offer a wide range of basic biological functions. The problem of understanding the complex biological behaviour induced by synthetic genome modifications can be addressed by the application of abstract models, therefore calling for mathematics and computer science. The development of a model of biological entities and processes is now considered an essential conceptual and computational tool for the design and analysis of biological systems [4]. The motivations are several, such as the support for accurate description of the components, evaluation of the alternatives, prediction of the time behaviour, study of causality relations between processes, simulation of in-silico experimentations and other kinds of reasoned analysis. A crucial point is the development of modelling frameworks that offer to modellers suitable primitives and features for a successful representation and investigation of the system.

Due to the similar scenarios of interests, it comes naturally for *Synthetic Biology* to turn to the modelling solutions proposed for the (relatively) older discipline of *Systems Biology* [5]. In the latter the systems that are investigated belong to a broader and less defined class of biological phenomena, but much research deals with the molecular cells' internal mechanisms that are of interest to *Synthetic Biology* modelling. In the *Systems Biology* context, (*stochastic*) *process algebra* formalisms have been proposed by several authors as valid modelling tools for molecular biological networks [6], [7]. The *Process algebra* proposition is sustained by the observation that the formalism supports both *qualitative* and *quantitative* analysis as well as primitives for a *modular* and *abstract* representation of entities and processes.

These characteristics highlight *process algebras* as an interesting candidate for the particular *Synthetic Biology* scenario but an explicit mapping between the two worlds is as yet lacking. This work therefore aims to establish a formal relation between *Synthetic Biology* constructs (composed of standard DNA parts) and *process algebra* modelling, in order to evaluate its suitability and to create a first tutorial base for further investigations.

1.1 Motivations

The work presented in this dissertation is motivated by the necessity of developing computational techniques for the representation and analysis of systems that are constructed using *standard biological DNA parts* defined in the Synthetic Biology discipline. The choice of a suitable framework for Synthetic Biology modelling must pass through an accurate comparison between the characteristics of the candidates and the particular need of the discipline. There are several reasons why process algebras seem to be valid candidates:

- **Formal:** the biological approach usually clarifies the fundamental aspects of systems but its often far from being formally defined. Instead, process algebra models are given as program-like source codes whose interpretation, according to defined syntax-driven rules (operational semantics), is exact and precise.
- **Qualitative:** biological systems contains intricate networks of entities and interactions whose structure is often difficult to represent and analyse. Having been originally developed for concurrency systems analysis, the process algebra formalism supports the representation and reasoning about complex interactions between components.
- **Quantitative:** biological investigation requires the ability to reason in reference to quantitative information. The expressiveness of process algebras permits the representation of populations of entities, thus supporting cardinality of elements. Moreover stochastic process algebras enable specification of the timing aspects of models.
- **Probabilistic:** there is a general consensus that stochastic and noise effects are essential building elements of biological mechanisms, in particular for *gene regulatory networks*. Stochastic process algebras map models to stochastic processes thus supporting probabilistic representation and reasoning.
- **Abstraction:** in general, when constructing a model there is the need to avoid unuseful or complex details. For the Synthetic Biology approach this is a central aspect and the different levels of details have been formalised in the *abstraction hierarchy* [3] for biological systems. Process algebras have few but powerful operators that represent basic actions and elements, thus permitting the modellers to tune the level of detail of the representation.

- **Compositionality:** biological systems are composed of entities and often experimentation relies on the ability to remove or add individuals, or merge separate functional modules. Using process algebras it is possible to first design individual components and then construct modules as their union, thus supporting compositionality.
- **Tools:** for answering biological questions the systems might need to be observed using different approaches. Since from its use in computer science, process algebras have been equipped with steady-state and transient model analysis tools, as well as applications for simulation and (probabilistic) model checking of models.
- **Experience:** biological mechanisms are often not completely understood or difficult to characterise qualitatively, leaving to the modeller the problem of identifying the right modelling *abstraction*. Experience gained from previous work in the area can be of immense help in dealing with new situations. In the recent past variants of stochastic process algebras have been used for modelling several types of molecular biological systems such as *metabolic reactions*, *signalling pathways*, *compartments structures* and *regulatory gene networks*.

1.2 Objectives and methods

The purpose of this work is to show the suitability of the process algebra formalism for the modelling of Synthetic Biology systems. The intent is mainly tutorial and seeks to highlight the possibility offered from the approach by developing the theoretical basis of a first small workbench. The objectives of the work can be summarized as follows:

- **Formal mapping:** to establish a formal mapping between systems constructed using DNA parts taken from the *Library of Standard Biological Parts* [3] and stochastic process algebra models.
- **Representative:** to undertake a modelling approach that uses fundamental shared characteristics among the process algebra family in order to be representative for the formalism class. Moreover to enable access to the most important kind of reasoning tools to show the full analysis potential.
- **Tutorial:** to present real modelling examples of the most common biological scenarios in Synthetic Biology.

- **Theoretical:** to discuss the physical, biological and mathematical assumptions underlying the application of the formalism and its analysis tools.
- **Implementable:** to present the work formally, using pseudo-code definitions in order to provide precise implementation guidelines for a future realization of the workbench.

The steps that were identified in order to achieve the objectives are as follows:

- **Graphical notation:** to develop a formal graphical notation for representing systems that are composed of standard biological parts (taken from the *Registry of Standard Biological Parts*). Consider the most important elements and relations, explain their biological meaning and define their representation as a structured data type.
- **Automatic translation:** to develop an automatic translation from the graphical notation to process algebra models. To use the typed structure of visual objects to implement an automatic derivation of the components and rules to construct the global systems's model.
- **PEPA:** to use the Performance Evaluation Process Algebra (PEPA) language as representative of the process algebra class. PEPA was selected among others because of its concise but powerful syntax composed of few operators, the ability to handle multi-way synchronizations, some previous interesting works related to the topic and supervisor expertise.
- **PRISM:** to develop an automatic translation from PEPA models to PRISM models, in order to enable analysis with *stochastic simulation*, *steady state* and *transient* analysis and *probabilistic model checking*.
- **Examples:** to identify the most important classes of biological networks in Synthetic Biology and show how the modelling approach can be used to model their fundamental biological mechanisms and patterns. To test the defined modelling approach on a real case study.

1.3 State-of-the-art

In *Synthetic Biology*, the most widespread approach to modelling is without doubt based on *ordinary differential equations* (or ODEs). Due to ODEs versatility in rep-

representing time and space properties of dynamical systems, the formalism has been widely adopted in modelling the main aspects of molecular biological systems, such as *gene regulatory networks*, *metabolic networks* and others. The models are generally in the form of a set of reaction-rate equations expressing the dependency of an entity's quantity, i.e. population of protein, as a function of other quantities present in the system. The main quantities are in general depending on time (and sometimes on space). By solving the set of equations it is possible to study the time behaviour of the system. Because of the usual non-linear nature of the functions, analytical solutions of the equations are often not possible and they need to be solved by numerical integration methods with approximations of the infinitesimal time. A major objection against the ODE approach is the representation of systems as a *continuous* and *deterministic* process. These assumptions may be questioned at the molecular level, where physical mechanisms are thought to be *discrete* and *probabilistic*. Another criticism concerns the inherent difficulty in modifying the model because of the poor compositionality of mathematical syntax.

In order to account for probabilistic aspects and discrete quantities, some stochastic alternatives have been developed. *Stochastic differential equations* are differential equations in which some of the terms are stochastic processes. This approach is a variant of the ODE method in order to count for randomly distributed events and can be used to model discrete quantities. Unfortunately in this framework model construction is quite a specialist process, in which complex stochastic effects are difficult to build and the resulting set of equations is often time-consuming to simulate.

Other stochastic approaches are in general based on the existence of a *Master Equation*, an equation that describes the time evolution of the probability to occupy one state from a discrete set of states. Although providing a rigorous stochastic account of the system behaviour, the Master Equations is usually very difficult to be solved [8].

For this reason, practical analysis usually relies on directly simulating the stochastic time evolution of the system, calculations that are termed *stochastic simulations*. They are usually fairly simple algorithms that in time update the system's state according to the probability of leaving the current state. Investigation of the models can then be calculated as statistical analysis of a sufficient number of simulations. The first application of stochastic simulation to biological systems was developed by Gille-

spie (see Section 2.6 for details) in the context of biochemical reaction systems and is based on a rigorous derivation of the Chemical Master Equation (CME). Although in general not supported by a similar firm physical derivation, stochastic simulations have been widely applied to other kinds of biological mechanisms, i.e. membrane networks. In this approach models are usually expressed using the notation proper of chemical reactions with interacting entities as ‘reagents’ and resulting entities’ modifications as ‘products’. The drawbacks of such a simple notation is the difficulty in complex model design, along with poor support for compositionality and abstraction.

1.4 Structure of the dissertation

Including this introductory chapter which also surveys the-state-of-the-art situation, the dissertation consists of six chapters. The next chapter reviews the background material needed to understand this work. Chapter 3 proposes the graphical notation that has been developed in order to represent Synthetic Biology systems. Chapter 4 presents rules for the automatic conversion from graphical representations of systems to PEPA models and highlights the modelling assumptions. Chapter 5 discusses the application of the modelling approach within a wide class of molecular biological networks and presents examples of some common scenarios. Finally, in Chapter 6 conclusions are drawn and future directions of the work are discussed.

Chapter 2

Background

This chapter presents the background material needed to understand and place in context the work. Section 2.1 presents an overview of the most important aspects of *Synthetic Biology* discipline. Section 2.2 proposes a brief review on the use of stochastic process algebra formalism for biological modelling in the discipline of *Systems Biology*. Section 2.3 introduces the process algebra *PEPA* and Section 2.4 presents the characteristic of the probabilistic model checker software tool *PRISM*. In Section 2.5 is introduced and explained the concept of *Continuous Time Markov Chain* stochastic process. Finally in Section 2.6 is presented the Gillespie's algorithm, used for the simulation of biochemical reaction systems.

2.1 Synthetic Biology

The term *Synthetic Biology* was firstly introduced in the 80's in the context of recombinant DNA technology to describe bacteria that had been genetically engineered and therefore were not anymore natural, but synthetical, in their DNA arrangement [1]. During the last decade, the term has found new life with reference to efforts to 'redesign life', and now it identifies the growing research area that deals with the engineering of biological systems that do not exist in nature and the re-engineering of existing biological ones.

Due to the complexity of such tasks, Synthetic Biology has become one of the first research fields in which strong multidisciplinary is needed and in which the goals vary widely with the respect of the groups' interests. Biologists seek to test their understanding of natural biological systems through the design and construction of synthetic replications. Chemists aim to exploit the capability of novel molecules and

molecular systems to develop diagnostic assay and drugs, to name just two examples. Engineers consider biology as a technology and seek to combine a broad expanse of biotechnology applications with the development of fundamental technologies [3]. For computer scientists, Synthetic Biology research is a generator of a huge amount of data that need to be stored, reasoned and shared, and systems that need to be designed, modelled and predicted. Due to the novelty of the field and the complexity of biological systems, it is really difficult to trace the borders of Synthetic Biology by defining how a synthetic biological system is constructed. The most used procedure is the transformation of organisms, called hosts, by the insertion of plasmids containing genetic recombinant DNA or the modification of (parts of) their original chromosomes, using recombinase techniques. The introduced sequence exploits the genetic machinery of the host and causes the occurrence of unnatural processes of transcription, translation and proteins interactions with host pathways. The physical design and realization of synthetic biological systems thus relies on the application of state-of-art gene recombinant technologies, the fundamental ones being:

- *DNA Sequencing*: are a set of methods for determining the order of the nucleotide bases (sequence) in DNA oligonucleotide. Since DNA sequences contain the genetic instructions used in the development and functioning of living organisms, the ability to ‘read’ it permits the creation of libraries of functional parts.
- *Restriction enzymes*: are enzymes able to cut double-stranded DNA by making two incisions at the site of a specific DNA sequence. They are used to extract existing functional DNA sequences from living organisms. The DNA pieces are usually left with ‘sticky ends’ that, in presence of ligase, enzymes that catalyse the joining of complementary strands, permit the composition of new sequences.
- *Polymerase Chain Reaction (PCR)*: is a technique which is able to amplify a double-stranded DNA oligonucleotide by performing an *in vitro* enzymatic replication. By applying PCR technique to a few copies of DNA pieces it is possible to replicate them to several orders of magnitude. This fundamental method that can be used to amplify DNA fragments containing any interesting DNA sequence, for example containing genes, promoters, binding regions and so on.

On the top of these technologies, the Synthetic Biology approach proposes and defines engineering paradigms in order to increase the reliability, efficiency and re-

usability of system construction. Borrowed from the past engineering experience emerged from other natural sciences, there are three main ideas that seem most relevant and form the ‘*Foundations for engineering biology*’ [3]:

- *Abstraction*: is a theoretical tool for managing the complexity of concepts and objects by retaining only the information relevant for the particular purpose. Since natural biological systems are thought to be complex on several aspects (huge amount of objects, different levels of view, non-linear interactions), it becomes evident the usefulness of an approach able to simplify their handling.
- *Standardization*: is an aspect that facilitated modern life in several fields and consists mainly in the agreement of individuals on definitions, protocols and methodology. In the context of Synthetic Biology is proposed to “support the definition, description and characterization of the basic biological parts, as well as standard conditions that support the use of parts in combination” [3]. It also permits an easier sharing of information and methodology of work between research teams and thus helps the creation of a sustainable community.
- *Decoupling*: is a paradigm stating that it is useful to separate bigger problems in many independent simpler ones in order to exploit specialization and expertise on individual tasks and then combine the intermediate results to obtain the final outcome.

The application of these three paradigms by the community of Synthetic Biology researchers have found notable results in the formalization and construction of the *Registry of Standard Biological Parts*, the development of the *Biobrick*TM concept, the definition of an *abstraction hierarchy* for biological systems and the decoupling between DNA *design* and *fabrication*.

The *Registry of Standard Biological Parts* (Figure 2.1), hosted by the MIT, is a free accessible database that provides a list of basic biological functional parts that can be used to construct synthetic biology systems. The parts are stored according to functionalities and abstraction hierarchy layers and are furnished with information regarding design consideration, source organism, work experience and other relevant information. Moreover all parts are physically stored in the form of standard BiobricksTM on well source plates and thus ready to be used in wet laboratories. The registry is maintained by the community on a free-open policy that allows everyone to contribute and it is enriched with help pages, software tools and support to communication.

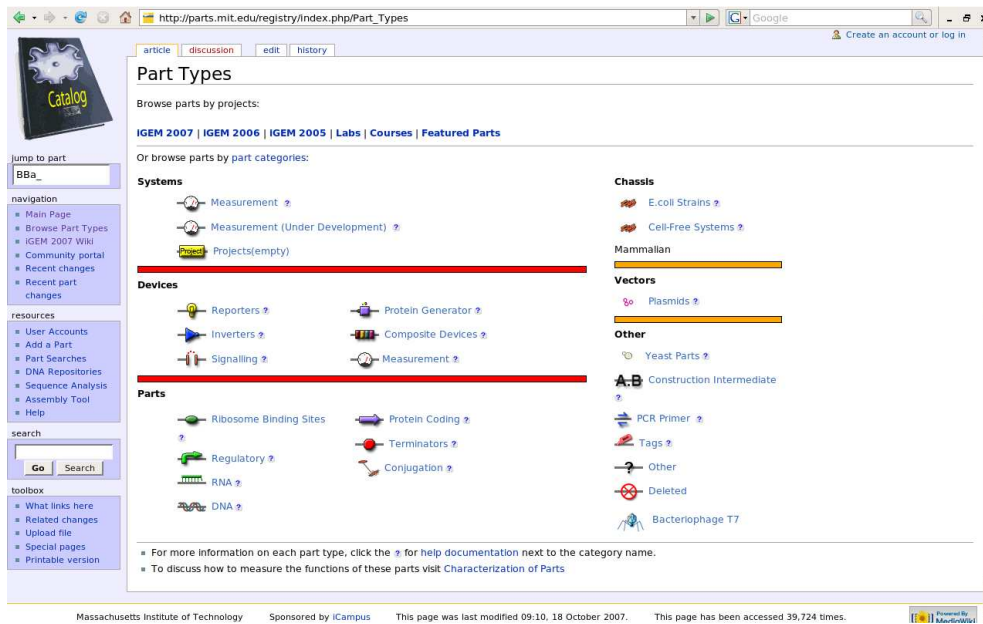


Figure 2.1: Screenshot of the Registry of Standard Biological Parts (<http://parts.mit.edu>): the main page with the hierarchical divisions of parts.

BiobricksTM[9] is a standard for interchangeable parts, developed in order to facilitate the construction of complex biological parts and based on the concept of compositionality: parts are compatible so that they can be connected in any order and in any number, still forming a BiobrickTM[9]. The concept relies on a formalized structure of suffix and prefix in the DNA sequence so that the parts can be assembled in labs through a process called *Standard Assembly*.

In order to manage the complexity of the biological systems, an abstraction hierarchy has been defined with the aim of allow individuals to work independently at each level of the hierarchy (Figure 2.2). The current structure presents four layers: *DNA*, *Parts*, *Devices* and *Systems*. Each layer focuses on different aspects: the *DNA* layer regards genetic material, the *Parts* layer refers to basic biological functions, the *Device* layer concerns composition of parts having standard input and output signals and the *System* layer identifies systems made of close functional device composition. Layers communicate through interfaces that limit and specify the exchange of information.

Notable results related to Synthetic Biology have been achieved in topics such as drug discovery and production, construction of engineering devices and biological computation [10],[11].

The annual *International Genetically Engineered Machines (iGEM)* competition, held at the MIT, is a worldwide competition that challenges (under)graduate students

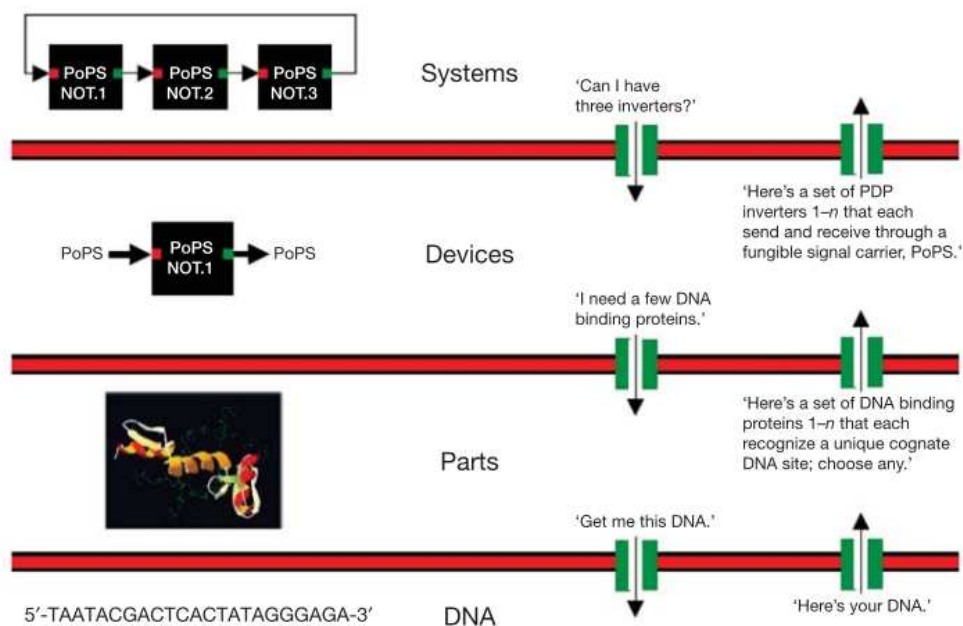


Figure 2.2: The abstraction hierarchy that supports the engineering of integrated genetic systems. From [3].

in the design and construction of Synthetic Biology systems. Provided with the Registry of Standard Biological parts, the teams from various Universities try practically to prove the possibility to “enable the systematic engineering of biology, promote the open and transparent development of tools for engineering biology and to help construct a society that can productively apply biological technology” [12].

2.2 Stochastic Process Algebras in Systems Biology

The *Systems Biology* [5] discipline aims to get an insight into the system-level of biological networks by developing suitable methodologies and paradigms of investigation. Over the last ten years, in the context of *Systems Biology*, formal techniques developed in computer science have gained increasing attention as conceptual tools for modelling analysis. Among them, the family of *stochastic process algebras* (or *stochastic process calculi*) has been proposed as a sound qualitative and quantitative framework for the representation and analysis of complex biochemical structured systems, such as the cell. Much, if not all the research, is driven by the ‘cell-as-computation’ approach [13], in which the biological elements are modelled as computable entities called *processes* (see Table 2.1 for the mapping). Processes are formal descriptions with internal

states and interaction capability. Their interactions can lead to internal state changes that represent the system's dynamic behaviour.

Biology	Process Algebra
Entity	Process
Interaction capability	Channel
Interaction	Communication
Modification-Evolution	State Change

Table 2.1: The process algebra abstraction for Systems Biology.

Several characteristics of process calculi motivated their application in modelling biological systems. Process algebra is a formalism with well-defined syntactical and semantics rules, therefore the interpretation of models is unambiguous. The language is usually composed of few but powerful operators and this permits to modellers to easily change the level of detail of the components, thus supporting *abstraction*. Components can be specified separately and then joined in a modular component, thus supporting *compositionality*. In their representation of systems they can handle quantitative information such as stochastic time behaviours and cardinality of entities. Moreover they are equipped with a whole range of tools for model reasoning such as equivalence relations, model checking and stochastic simulation. Languages that were adapted or extend from languages original conceived for distributed systems modelling, such as π -calculus, PEPA, BioAmbient and CCS-R, are termed as *bottom-up* calculi [14]. They can rely on past experience and tools but usually lacks of specialization for handling specific biological details. Other process algebra *dialects* have been developed in recent years in order to specialize on particular biological aspects, they are termed *top-down* languages [14]. Since it was the first area of application, in general all algebras support the representation of *biochemical reaction networks*. Few of them such as *Brane Calculus* and *BioAmbient* propose extensions for dealing with *compartments*: the former focused on computation on membranes, the second on the location of molecules in specific compartments. The *Beta-Binders*, inspired by enzyme theory, introduces the concept of interface thus permitting the interaction of entities not completely matching but affine. Interesting applications of process algebras can also be found in the modelling of *metabolic networks* [15]. *Gene regulatory networks*, fundamental actors in Synthetic Biology systems, were not studied specifically until recently in the works of Cardelli et al. [16],[17].

2.3 PEPA: process algebra for systems modelling

Process algebras are formal languages that have been widely used in the last thirty years to model various types of concurrent systems, especially in the context of communicating and interacting processes. The first process algebras developed in the 70's, such as *Calculus of Communicating Systems* (CCS)[18] and *Communicating Sequential Processes* (CSP) [19], proved the usefulness of a structured and firm mathematical approach for reasoning about models' behaviour. Afterwards, based on the same core principles, several different process algebra formalizations have been proposed with the aim to refine the expressional power of the language or enhance it with additional analysis features. In this category we find the family of *Timed Extended Process Algebras*.

Since in pure process algebras the time aspects of models are abstracted away by assuming an instantaneous realization of events, these process algebras enhanced the language by adding duration attributes to their actions. In this family, Hillston's *Performance Evaluation Process Algebra* (PEPA) [20], was the first to associate with each action a stochastic duration according to an exponential distribution. This leads to the possibility of deriving from each process algebra system definition a mathematical model in the form of a *Continuous Time Markov Chain* (CTMC) and thus to perform a qualitative and quantitative set of analyses.

In this section we introduce PEPA by presenting its syntax and semantics rules as well as an overview of its tools and properties. We also discuss its relation to CTMCs in the context of systems analysis.

The original work on PEPA was motivated by the need for performance analysis in large communication systems and thus the expressiveness of the language is focused on parallel behaviours and synchronized activities. PEPA systems are represented as a collection of components able to perform activities and that can be composed by using operators such as cooperation and choice. Each activity is assumed to have a stochastic duration that is defined by a random variable with an exponential distribution. The syntax of PEPA components, called processes, P is formally defined as:

$$P = (\alpha, r).P \mid P + Q \mid A \stackrel{\text{def}}{=} P \mid P \underset{L}{\bowtie} Q \mid P/L$$

Prefix: $(\alpha, r).P$. This is the basic term by which components are constructed. It defines a component which can perform an action of type α . The action has a stochastic

duration governed by an exponential distribution with parameter r . After having performed α the component behaves as P .

Choice: $P + Q$. This represents a component that can behave either as component P or Q and thus enables all concurrent activities of the two components. The component that first completes an activity is chosen and the other is discarded.

Constant: $A \stackrel{def}{=} P$. A constant is a component whose behaviour is defined as equal to an expression of already defined components. It allows us to associate names to pre-defined behaviour patterns.

Hiding: P/L . This states that the activities in L are internal and private to component P and thus cannot be synchronized or be seen by external components.

Cooperation: $P \bowtie_L Q$. This denotes the cooperation between components P and Q over the set of activities L . The set L lists the activities on which the two components are forced to synchronize. For activities outside L the components proceed in an independent and concurrent fashion. *Multiway cooperation* is supported, meaning that the activity resulting from a synchronization of two components can be synchronized again with another component. In the original PEPA definition, the synchronization rate of an activity is defined as the minimum rate of the synchronizing activities. For our purposes in the context of Synthetic Biology modelling (and in general for biochemical systems), PEPA semantics have been modified in order to calculate the synchronization rate as the product of the rates of synchronizing activities (see Figure 2.4).

The dynamic and formal interpretation of PEPA expressions is given by its structured operational semantics (see Figure 2.3).

By applying the structural operational semantics rules to a PEPA component definition P , it is possible to generate a representation of the system as a stochastic process in the form of a *Continuous Time Markov Chain* (CTMC). Firstly, the labelled transition system called the derivation graph is obtained. This is a multigraph in which each node is one of the possible subsequent derivatives of the component defining the model and in which nodes are connected by arcs representing the activities that lead from one component to the other. The arcs are labelled with the action type and activity rate and the root node is the component defining the model. In order to derive

<p>Prefix</p> $\frac{}{(\alpha, r).E \xrightarrow{(\alpha, r)} E}$ <p>Choice</p> $\frac{E \xrightarrow{(\alpha, r)} E'}{E + F \xrightarrow{(\alpha, r)} E'}$ $\frac{F \xrightarrow{(\alpha, r)} F'}{E + F \xrightarrow{(\alpha, r)} F'}$ <p>Hiding</p> $\frac{E \xrightarrow{(\alpha, r)} E'}{E/L \xrightarrow{(\alpha, r)} E'/L} \quad (\alpha \notin L)$ $\frac{E \xrightarrow{(\alpha, r)} E'}{E/L \xrightarrow{(\tau, r)} E'/L} \quad (\alpha \in L)$	<p>Cooperation</p> $\frac{E \xrightarrow{(\alpha, r)} E'}{E \bowtie_L F \xrightarrow{(\alpha, r)} E' \bowtie_L F} \quad (\alpha \notin L)$ $\frac{F \xrightarrow{(\alpha, r)} F'}{E \bowtie_L F \xrightarrow{(\alpha, r)} E \bowtie_L F'} \quad (\alpha \notin L)$ $\frac{E \xrightarrow{(\alpha, r_1)} E' \quad F \xrightarrow{(\alpha, r_2)} F'}{E \bowtie_L F \xrightarrow{(\alpha, R)} E' \bowtie_L F'} \quad (\alpha \in L)$ <p>where $R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$</p> <p>Constant</p> $\frac{E \xrightarrow{(\alpha, r)} E'}{A \xrightarrow{(\alpha, r)} E'} \quad (A \stackrel{def}{=} E)$
--	--

Figure 2.3: PEPA Structured Operational Semantics. Details in [20].

the stochastic process, a state is associated with each node in the graph and a transition between states is defined for each arc. Under the assumption that the model is finite it is proved that the generated stochastic process is a CTMC [20]. The CTMC underlying every PEPA model is time homogeneous and irreducible, and for such CTMCs, it is possible to calculate a equilibrium probability distribution. This probability gives information about the CTMC in the long run or steady state. From this probability distribution, it is possible to calculate performance measures of the model such as utilisation and throughput. The CTMCs can also be used to obtain other analyses such as transient and passage-time analyses, which allow study of the system at any instant of time before it reaches the steady state discussed earlier. The PEPA Workbench [21] is an available software tool that permits the automated analysis of PEPA generated CTMCs but unfortunately it implements the original PEPA semantics and thus could not be used for our model analysis. For this reason, and also for enabling probabilistic model checking, we developed a translation from PEPA models to PRISM [22] models

$$\begin{array}{c}
\textbf{Cooperation} \\
\frac{E \xrightarrow{(\alpha, r_1)} E' \quad F \xrightarrow{(\alpha, r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha, R)} E' \underset{L}{\bowtie} F'} \quad (\alpha \in L) \\
\text{where } R = r_1 * r_2
\end{array}$$

Figure 2.4: The modification to the PEPA cooperation rule in order to calculate the cooperation rate as the multiplication of the participant rates.

in order to use PRISM analysis support tool. Details of the probabilistic model checker PRISM are presented in Section 2.4.

2.4 PRISM: probabilistic model checker and simulator

PRISM is a probabilistic model checking tool used for analysing quantitative properties of systems which exhibit stochastic behaviour [22]. The tool offers facilities to “manually explore models, Monte-Carlo discrete-event simulation techniques for approximate model analysis (including support for distributed simulation) and the ability to compute cost- and reward-based measures” [23]. We are interested in PRISM because it offers the specification of *continuous time Markov chains* (details in Section 2.5) through a simple state-based language and enables their Monte-Carlo simulation, as well as analysis via temporal logic CSL with extensions for quantitative specifications and costs-rewards. The tool is equipped with a graphical user interface for the managing of models’ code, model checking queries and graph-plotting functionalities.

Operator	CSL Syntax	Operator	CSL Syntax
True	<i>true</i>	Next	$P_{\bowtie p}[\mathbf{X}\phi]$
False	<i>false</i>	Unbounded Until	$P_{\bowtie p}[\phi\mathbf{U}\phi]$
Conjunction	$\phi \wedge \phi$	Bounded Until	$P_{\bowtie p}[\phi\mathbf{U}^{\leq t}\phi]$
Disjunction	$\phi \vee \phi$	Bounded Until	$P_{\bowtie p}[\phi\mathbf{U}^{\geq t}\phi]$
Negation	$\neg\phi$	Bounded Until	$P_{\bowtie p}[\phi\mathbf{U}^{[t_1, t_2]}\phi]$
Implication	$\phi \Rightarrow \phi$	Steady State	$S_{\bowtie p}[\phi]$

Table 2.2: Continuous Stochastic Logic operators.

The logic CSL (Continuous Stochastic Logic) is used to define probabilistic formulae which validity can be checked, or probabilities can be computed, by the model

checker module implemented in PRISM. Using CSL is possible to express probability measures about temporal properties to be satisfied either in steady state or in transient behaviour. The operator $S_{\triangleright p}[\phi]$ specifies properties ϕ that hold in the long run (steady state), the operator $P_{\triangleright p}[\phi]$ defines properties ϕ that depend on time (transient). The value p specifies a probability bound on the property. Using the temporal logic is possible to express temporal queries on generic model characteristics such as *state reachability* or *deadlock*, or on specific characteristics representing particular aspects of the considered scenario. The operators of CSL are listed in Table 2.2, for a detailed discussion see [22].

We give here an informal and brief explanation of PRISM simple modelling language, for a more detailed definition see [23]. *Modules*, *variables* and *commands* (an example is shown in Figure 2.5) are the fundamental components of the PRISM language. A model is composed by global variable definitions and several modules. Each module contains a list of commands and a set of local variables. Variables are typed (they can be integers, reals and booleans). A command is composed of a name, a boolean guard, a rate specification and a set of instructions that may change variable values.

```

module A                                \name of the module
  x : [0..3] init 0;                    \local variable of integer type
  [action] (x<3)                        \command called "action" with boolean guard
  : x -> x'=x+1;                        \x is the rate, then variable x is incremented
endmodule                               \end of the module

```

Figure 2.5: A simple example of a PRISM module definition, with comments.

The values of global and local variables at any given time define the state of the model. When the boolean guard of a command is satisfied, the commands can synchronize with other enabled commands with the same name. This causes the execution of their instruction, possibly leading to a change of state in the system. An exponential rate, calculated as the multiplication of each single synchronizing command's rate, is associated with the enabled transition. Each model thus specifies a CTMC (see Section 2.5) as a direct mapping of model states and transition sets.

2.5 Continuous Time Markov Chains

As explained in Section 2.3, the stochastic models specified using PEPA generate a system representation in the form of a particular type of stochastic processes: *continuous time Markov chains* (or CTMCs). Formally, a stochastic process is defined as a set of random variables $\{X(t), t \in T\}$ where T is called the *index set*. The set of all possible values that variable $X(t)$ can assume is called the *state space*. Each of these values is called a *state* of the process. A Markov process is a process for which the *memoryless property* holds. Formally:

Definition $\{X(t)\}$ is a Markov Process if and only if:

$$Pr(X(t+h) = y | X(s) = x_s, \forall s < t) = Pr(X(t+h) = y | X(s) = x_t), \forall h > 0$$

This property reads as “the distribution of time until the next state change is independent of the time that has elapsed since the last state change”. CTMCs are Markov processes with a countable *state space* and for which the *index set* is assumed to represent a continuous time ($T = \mathbb{R}^+$). A Markov process is *time homogeneous* if the transition rates are independent of the time of occurrence of each transition, such that:

$$P(X(t+s) = j | X(t) = i) = P(X(s) = j | X(0) = i).$$

Our interest will be limited only to time homogeneous Markov chains. Because of the *memoryless property*, the probability of no transitions occurring from state i in some time r is governed by an exponential distribution of parameter q_i (called *exit rate*):

$$Pr(X(s) = i, \forall s \in (t, t+r] | X(t) = i) = q_i * e^{-q_i * r}$$

Thus the probability of an instantaneous transition from state i to state j is:

$$Pr(X(t+dt) = j | X(t) = i) = q_{ij} * dt + o(dt)$$

where q_{ij} is the *instantaneous transition rate* from state i to state j and $q_i = \sum_{j \neq i} q_{ij}$. CTMCs can be described by the infinitesimal generator matrix Q for which each element q_{ij} with $j \neq i$ is the transition rate between states i and j . Diagonal elements q_{ii} are defined as $-\sum_{j \neq i} q_{ij}$. Markov chains can be also described as a directed graph, where the edges are labeled by the exponential probabilities of going from one state to the other state. The infinitesimal matrix is used in the calculi for the transient and steady state analysis. The time dependent probability is the solution of the following differential equations, called Chapman-Kolmogorov equations:

$$\frac{d\pi(t)}{dt} = \pi(t) * Q$$

where the stationary probability distribution π , normalized to $\sum_j \pi_j = 1$, is the solution of the linear system $\pi * Q = 0$. The steady state probability distribution is the probability of being in a state in the long run. If a Markov chain is irreducible and positive recurrent, the steady state corresponds to the stationary probability. A Markov chain is irreducible if all states can be reached from all other states and is positive recurrent if starting in any state the expected time to return to that state is finite. In this case the stationary probability distribution is the solution of the linear system:

$$\lim_{t \rightarrow \infty} \pi(t) = \pi$$

In general the whole infinitesimal generator matrix Q is needed for such analysis but unfortunately for large and complex systems the underlying CTMC is too big to be computationally tractable. One possibility is to introduce an approximation and statistically study the behaviour of some samples of the process. A *sample path*, or *realization* of the process, is any set of instances of $\{X(t), t \in T\}$ and can be calculated using Monte Carlo algorithms without generating the complete CTMC state space.

2.6 The Gillespie's Algorithm

The *Gillespie algorithm* [24] is a stochastic exact procedure used to simulate the course time behaviour of spatial homogeneous chemical systems. It is a fairly simple *Monte Carlo algorithm* that numerically simulate the behaviour of molecule populations in a system of elementary chemical reactions. The chemical system is usually given in the form of M chemical reactions:

$$\text{Chemicalsystem} = \begin{cases} R_1 : c_1^1 X_1 + \dots + c_1^N X_N \xrightarrow{k_1} c_1^1 X_1 + \dots + c_1^N X_N \\ \dots \\ R_M : c_M^1 X_1 + \dots + c_M^N X_N \xrightarrow{k_M} c_M^1 X_1 + \dots + c_M^N X_N \end{cases}$$

X_1, \dots, X_N are N chemical species, R_1, \dots, R_M are M elementary reactions, k_1, \dots, k_M are M stochastic constants for the reactions, c_i^j are $N * M$ reagent stoichiometric coefficients (possibly with value zero) and c_i^j are $N * M$ product stoichiometric coefficients (possibly with value zero). The algorithm is a rigorous mathematical derivation of the *Master equation* and thus has firmer physical basis than the equivalent deterministic

formulation [24] with *Ordinary Differential Equations*. The *Master equation* identifies the evolution of the system as a *continuous-time Markov process*, with the integrated master equation obeying a Chapman-Kolmogorov equation. In this sense, the Gillespie algorithm calculates a *sample path* (see Section 2.5) of that CTMC without explicitly storing its often intractable state space. Each state in the CTMC derived from the Master Equation represents the current species populations and can be expressed as a vector $v = (X_1, \dots, X_n)$. The transitions from state to state represents possible reactions and are associated with an exponential rate calculated following the *mass action* law. The steps of the algorithm are [24]:

- *Step 0 (Initialization)*: the input is given as the stoichiometric coefficients for the reactions, the vector of the initial molecular populations $v_1 = (x_1, \dots, x_n)$ and the stochastic reaction constants k_1, \dots, k_M . Set the time variable t and the reaction counter n both to zero. Initialize the *random number generator*.
- *Step 1*. Calculate and store the M stochastic reaction rates (a_1, \dots, a_M) each one being the product of the molecular reactant combinations according to stoichiometry and the stochastic reaction constants for that reaction. Calculate a_0 as the sum of a_i .
- *Step 2*. Using the *random number generator*, calculate a random number π according to the probability density function $P_1(\pi) = a_0 * e^{(-a_0*\pi)}$, and a integer μ according to the probability density function $P_2(\mu) = \frac{a_\mu}{a_0}$.
- *Step 3*. Using the π and μ values obtained in step 2, increase t by π , and adjust the molecular population levels to reflect the occurrence of one R_μ reaction.
- *Step 4* if the time t is less than the maximal simulation time wanted, return to *Step 1*.

The *Gillespie's Algorithm* is of particular importance in the analysis of biochemical reaction networks, where often the state space of the stochastic process is intractable if not using simulation techniques.

Chapter 3

A graphical notation for Synthetic Biology

This chapter presents a graphical notation suitable for representing systems that have been constructed using *standard biological parts*, as defined in the Synthetic Biology approach. Section 3.1 explains which are characteristics considered in designing the notation. Section 3.2 presents the symbols of our graphical notation and their mapping with biological entities and processes. Section 3.3 specifies the way in which symbols can be composed and Section 3.4 gives additional system's constraints.

3.1 Design objectives

In this chapter we introduce a graphical notation that can be used for describing systems constructed using Synthetic Biology standard parts. A graphical notation is useful because it helps us to represent, in a compact and intuitive format, complex biological elements and phenomena. In order to be effective, we seek to define a graphical notation with the following design objectives:

- **Intuitive:** The notation should be intuitive to a reader with a biology background. This means it should represent the entities and the interactions using symbols that are common in biology literature.
- **Quantitative:** The notation should carry quantitative information regarding time-course aspects and cardinality of entities involved, i.e. size of the molecular populations.

- **Formal:** The notation should be consistent with some formal rules in order to be used as input for an automatic derivation of the corresponding model.
- **Limited:** it should be limited to few but expressive symbols, able to represent the systems at the level of *Parts* (according to the definition in the *Registry of Standard Biological Parts*).

There is a general agreement that a standard graphical notation for biological system definition should be adopted to facilitate the understanding of drawings, and some works are currently investigating promising solutions [25], [26]. But a unified convention has not yet emerged from the research community and so in the literature it is common to find very different notations. Nevertheless, some symbols are generally well recognised in the scientific community and are a sort of *de-facto standard*. The graphical notation we propose here merges together two of these *de-facto standards*. For the definition of chemical reactions, we propose the *circles-connections-and-boxes* notation used by computer scientists working with *Petri Nets*. For promoters, coding regions, promoter repression and activation, we used the *shapes-and-arrows* notation usually found in bioengineering for representing DNA sequences and regulatory effects. Among many other *dialects*, we propose the version used in the *Registry of Standard Biological Parts* (see Section 2.1), as it should be very intuitive to the Synthetic Biology community.

We add alphanumerical and numerical attributes to the symbols: qualitative and quantitative information that are used to correctly manage time course behaviour and cardinalities when translating the model into the process algebra formalism.

Along with the graphical symbols and their biological explanation, we introduce a formal structured definition using the concepts of *type*, *attribute* and *object*. Their meaning should be quite obvious to computer scientists, we give here a brief definition. In our context a *type* is a name that defines the set of values that one *object* can assume. There are three primitive types: the type *Id* with values in the set of alphanumeric identifier, the type *Real* which allows values in the positive real numbers (\mathbb{R}^+) and the type *Nat* which allows values in the natural positive numbers (\mathbb{N}^+). We can also define *structured types* (or *records*) by giving a list of attribute names and types in between brackets, as in Figure 3.1. Structured type names begin with the letter *T* and are written in italics. An *object* is associated with a *type* by the colon operator, *object:Type*. This formal definition can be also of help as precise low-level pseudo-code for a future data structure implementation. We want to stress one point: the


```
TypeName: {  
  AttributeName1 : Type1;  
  ...  
  AttributeNameN : TypeN;  
}
```

Figure 3.1: A tutorial example of structure of a type definition.

graphical notation we propose here is not meant to be a generally applicable notation for all possible *Synthetic Biology* systems. Due to the variety and complexity of those systems, designing a notation with such a goal is a huge task. We seek only to define a limited exact notation that permits us to show that it is possible to generate process algebra model templates from an intuitive graphical schema.

3.2 Symbols

In this section we introduce the nine symbols that are unitary elements of the graphical notation. For each element we present its graphical representation and discuss the biological class that it represents in the context of Synthetic Biology. Moreover, we associate with each symbol a formal definition by creating a new *Type*. The symbols are divided in three broader categories: symbols used to represent reactions (Table 3.1), symbols used to represent DNA parts taken from the Registry of Standard Biological Parts (Table 3.2), symbols used to represent interactions between species and DNA parts (Table 3.3).

SPECIES: a species (Table 3.1) is a biological entity that can be used as reagent and/or product in a reaction, like a protein, an enzyme, a molecule or a currency metabolite. With each species is associated an alphanumeric identifier (attribute *id*). In the models the species' molecular population is constrained by a lower bound (attribute *min*) and an upper bound (attribute *max*). The initial population is given by the attribute *init*.

REACTION: the reaction (Table 3.1) is an abstraction for whatever interactions between species that behaves accordingly to the mass action law. Typical reactions in which we are interested are metabolite transformation, phosphorylation and dephosphorylation, cleavage, complex formation, enzymatic reaction and many more.

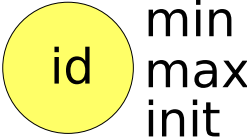
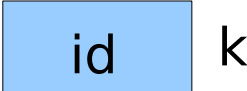
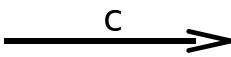
a)		<i>TSpecies:</i> { id : <i>Id</i> ; max: <i>Nat</i> ; min: <i>Nat</i> ; init: <i>Nat</i> ; }
b)		<i>TReaction:</i> { id : <i>Id</i> ; k: <i>Real</i> ; }
c)		<i>TReactionPart:</i> { c : <i>Nat</i> ; }

Table 3.1: Symbols that are used for the representation of reactions, with their formal definition. a) *Species* object. b) *Reaction* object. c) *Partecipation in a reaction* object.

Each reaction is associated with an alphanumeric identifier (attribute *id*) and a stochastic reaction constant (attribute *k*).

PARTICIPATION IN A REACTION: the participation arrow (Table 3.1) is used to connect species with reactions, and is used to define which are the inputs and which are the outputs. Since a reaction could require more than a single molecule of a species in order to happen, the attribute *c* indicates the stoichiometric coefficient for the species.

PROMOTER: the promoter symbol (Table 3.2) represents a regulatory element involved in the initiation of the process of transcription of the DNA in RNA. It can be constitutively active or inactive and it could be regulated by transcriptional factors that can repress or excite its activity. The attribute *e* is the rate of constitutive expression of the protein coding regions placed in its downstream region. The attribute *id* is an univoque alphanumeric identifier. The role of the promoter in the graphical notation is to control the expression of rows of downstream coding regions. It can be influenced by activation and repression effectors binding.

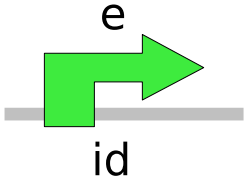
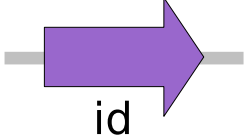
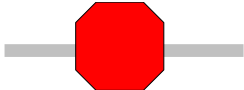
a)		<i>TPromoter:</i> { id : <i>Id</i> ; e : <i>Real</i> ; }
b)		<i>TProteinCod:</i> { id : <i>Id</i> ; }
c)		<i>TTerminator:</i> { }

Table 3.2: Symbols that represent DNA functional parts, with their formal definition.
a) *Promoter* object. b) *Protein Coding Region* object. c) *Terminator* object.

PROTEIN CODING REGION: a protein coding region box (Table 3.2) represents a piece of double strand DNA that codes for a specific functional protein. It can be placed downstream to a promoter or another protein coding. The attribute *id* is its alphanumeric identifier.

TERMINATOR: the terminator symbol (Table 3.2) represents a double-strand DNA segment that causes the end of the transcription process when encountered by the transcribing polymerase. It has no attribute since it serves only to graphically close the coding region sequence.

CODING CONNECTION: the coding connection arrow (Table 3.3) is used to connect a coding region to a species in order to specify that the region codes for that species. It has no attributes because it is simply a graphical connector to the species object.

ACTIVATION CONNECTION: the activation arrow (Table 3.3) is used to connect a species to a promoter in order to specify that the species can bind the promoter region and therefore enhance its expression rate. The two attributes of the arrow are: the rate of binding between the transcriptional factor and the promoter *b*, and the rate of additional expression *a* to the constitutive expression of the gene.




a)		$TCodingConn:$ { }
b)		$TActivationConn:$ { $b : Real;$ $a : Real;$ }
c)		$TRepressionConn:$ { $b : Real;$ $r : Real;$ }

Table 3.3: Symbols that represent interactions between DNA parts and species, with their formal definition. a) *Coding Connection* object. b) *Activation Connection* object. c) *Repression Connection* object.

REPRESSION CONNECTION: the promoter repression arrow (Table 3.3) is used to connect a species to a promoter in order to specify that the species can bind the promoter region and therefore decrease its expression rate. The two attributes of the arrow are: the rate b of binding between the transcriptional factor and the promoter, and the rate a at which the expression of the constitutive expression is repressed.

3.3 Composition rules

The symbols defined in Section 3.2 are the basic elements of our notation. In order to express biochemical interactions, they should be composed. To avoid the creation of connections without a biological meaning, a set of rules defining the allowed connections among the symbols given above is defined.

REACTION INPUT: a species (attribute S) can be connected using a reaction participation arrow (attribute Rp) to a reaction (attribute R), see Table 3.4. It means that the species is a reagent for the reaction and that, when an individual reaction occurs,

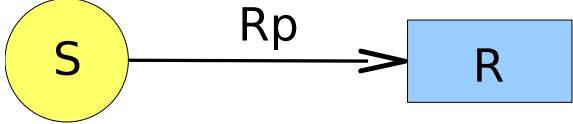
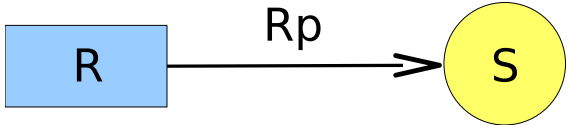
a)		<i>TReactionInput:</i> { <i>S</i> : <i>TSpecies</i> ; <i>Rp</i> : <i>TReactionPart</i> ; <i>R</i> : <i>TReaction</i> ; }
b)		<i>TReactionOutput:</i> { <i>S</i> : <i>TSpecies</i> ; <i>Rp</i> : <i>TReactionPart</i> ; <i>R</i> : <i>TReaction</i> ; }

Table 3.4: Patterns for the compositions of symbols in representing reactions, with their formal definition. a) *Reaction Input* object. b) *Reaction Output* object.

the species population decreases by a number of elements equal to the stoichiometric coefficient defined on the arrow.

REACTION OUTPUT: a reaction (attribute *R*) can be connected using a reaction participation arrow (attribute *Rp*) to a species (attribute *S*), see Table 3.4. It means that the species is a product for the reaction and that, when a individual reaction occurs, the species population increases by a number of elements equal to the stoichiometric coefficient defined on the arrow.

OPERON STRUCTURE: the term *operon* identifies a DNA structure in which a set of *n* coding regions (attributes Pc_1, \dots, Pc_n) are sequentially placed downstream of a promoter region (attribute *P*), see Table 3.5. Since the mRNA-polymerase binds to the promoter region, the expression of the coding regions is controlled by the promoter state. A gene can be represented as an operon with a single coding region.

PROTEIN EXPRESSION: a protein coding region (attribute *Pc*) is connected to a species (attribute *S*) by a coding connection (attribute *Cc*), see Table 3.5. It represents the fact that the coding region codes for that particular species and thus when the region is expressed the species population is replenished.

PROMOTER ACTIVATION: a species (attribute *S*) can be connected using a pro-

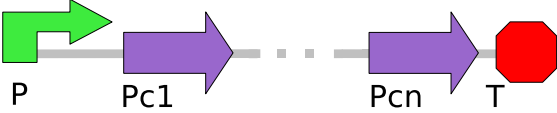
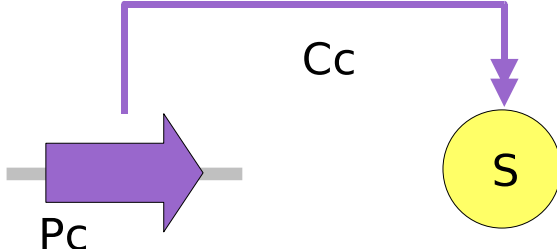
a)		<p><i>TOperon:</i></p> <pre>{ P : TPromoter; Pc1 : TProteinCod; ... Pcn : TProteinCod; T : TTerminator; }</pre>
b)		<p><i>TProteinCodExp:</i></p> <pre>{ Pc : TProteinCod; Cc : TCodingConn; S : TSpecies; }</pre>

Table 3.5: Patterns for the composition of symbols in representing operon expression, with their formal definition. a) *Operon Structure* object. b) *Protein Expression* object.

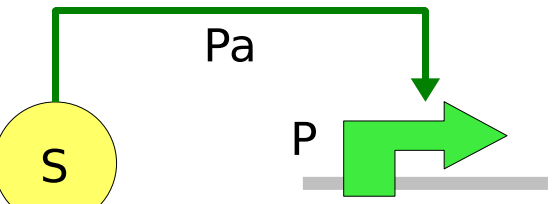
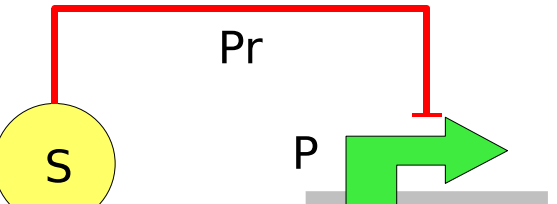
a)		<p><i>TPromoterAct:</i></p> <pre>{ P : TPromoter; Pa : TActivationConn; S : TSpecies; }</pre>
b)		<p><i>TPromoterRep:</i></p> <pre>{ P : TPromoter; Pr : TRepressionConn; S : TSpecies; }</pre>

Table 3.6: Patterns for the compositions of symbols in representing transcriptional regulation effects, with their formal definition. a) *Promoter Activation* object. b) *Promoter Repression* object.

moter activation connection (attribute *Pr*) to a promoter (attribute *P*), see Table 3.6. The activation happens as the result of a physical mechanism of binding of the transcriptional factors (the species' molecules) to the promoter region, attracting the mRNA-polymerase. Thus the transcription of the downstream regions is hastened. This can

lead to a speed up of the expression process.

PROMOTER REPRESSION: a species (attribute S) can be connected using a promoter repression connection (attribute Pr) to a promoter (attribute P), see Table 3.6. The repression happens as the result of a physical mechanism of binding of the transcriptional factors (the species' molecules) to the promoter region, preventing the binding of mRNA-polymerase. Thus the transcription of the downstream regions is blocked for the time the molecule is binded. This can cause a delay or a complete shut down of the expression process.

3.4 System's definition and constraints

A system is defined by the set of its elementary objects (instances of symbols) and the set of connection objects (instances of patterns). The elementary objects are specified following rules given in Section 3.2. The connection objects are specified following rules given in Section 3.3. Formally, a system Sys is defined as a pair of sets (**Elements**, **Connections**) where:

- **Elements** is a set of objects of type $TSpecies$, $TReaction$, $TReactionPart$, $TProteinCod$, $TPromoter$, $TCodingConn$, $TActivationConn$, $TRepressionConn$ and $TTerminator$ which represent the elements in the system.
- **Connections** is a set of objects of type $TReactionInput$, $TReactionOutput$, $TOperon$, $TProteinCodExp$, $TPromoterRep$, $TPromoterAct$ where the attributes of these objects are taken from the **Elements** set according to type definition.

In order to be sound a system should respect the following additional constraints:

- All the identifier values of type Id are different. This means that it is possible to use the identifier attributes as unique names for these elements.
- Each object in the **Elements** set should appear at least once as an attribute of an object in the set **Connections**, meaning that there cannot exist completely disconnected elements.
- An object of type $TProteinCod$ in **Elements** should be once and only once an attribute of an instance of type $TOperon$ in **Connections**. Meaning that a protein coding region can be regulated by only one upstream promoter.

- An object of type *TPromoter* in **Elements** should be once and only once an attribute of an object of type *TOperon* in **Connections**, meaning that a promoter can regulate only one set of coding regions.
- An object of type *TSpecies* in **Elements** and an object of type *TReaction* in **Elements** can be only once attributes together of an object of type *TReactionInput* in **Connections**, meaning that the participation of more individuals of a species as reagents in a reaction should be represented by using the stoichiometric coefficient.
- An object of type *TSpecies* in **Elements** and an object of type *TReaction* in **Elements** can be only once attributes together of an object of type *TReactionOutput* in **Connections**, meaning that the participation of more individuals of a species as product in a reaction should be represented by using the stoichiometric coefficient.

The rules we have given above should make intuitive the passage from the graphical specification of a system to its formal definition and vice versa. From the graphical definition it is possible to construct the sets of the formal definition by populating instances of the correct type and by composing the *Conn* set according to the graphical connections. From a formal definition it is possible to follow the picture examples in order to draw the correct symbols and to connect them. The two formulations are indeed carrying the equivalent amount of information. In Section 4 we show how to derive a process algebra model from the system's formal definitions.

Chapter 4

The PEPA model

This chapter proposes an automatic translation from a system, represented in the graphical notation for Synthetic Biology (see Section 3), to a model expressed using the process algebra PEPA (Section 2.3). Section 4.1 explains the concepts at the base of the modelling approach. Section 4.2 defines the formal rules for the translation of components and their composition in the final PEPA model. Section 4.3 discusses the properties of the resulting stochastic process with respect to biological mechanisms. Finally Section 4.4 explains how to automatically convert PEPA models into PRISM models in order to enable various type of system analysis.

4.1 A quantitative mechanistic approach

The biological systems we seek to model and analyse are cells. In general the assignment of a model to a system is not unique. Models can be different for various motivations: the choice of alternative mathematical frameworks, the study of different biological instances, the need to highlight different aspects of the same instance or the application of one particular experimental method of investigation [27].

We utilise the process algebra PEPA as an intermediate modelling language in between the graphical notation and the mathematical framework of *continuous time Markov chains* (see Section 2.5). We decided to undertake a mechanistic approach, meaning that we generate models whose internal descriptions match the molecular mechanisms by which biological processes act or are supposed to act [4]. Such a mapping is necessary when the aim is not only to predict and study the system behaviour, but also to use the comparison with lab experiments as feedback for refining our understanding of the real processes, i.e. for hypothesis testing support. In this way it is

also possible to incorporate quantitative knowledge obtained from wet lab experiments, such as the number of objects involved or the temporal rates at which the entities react. In order to be sound with respect to real processes, our model needs to be consistent with the known physical laws. We aim to model at a system level and considering the current body of knowledge in molecular and cell biology, this “system-level understanding has to be grounded onto molecular-level so that a continuous spectrum of knowledge can be established” [27]. The minimal entities that are explicitly represented are the populations of chemical species that can be micro or macro molecules such as proteins, carbohydrates, lipids and so on. We represent the DNA level in functional modular parts, dissociating *promoter regions* from *protein coding regions* and establish the rules for their composition into *functional operon regions*. The biochemical processes that we model are elementary chemical reactions between species, gene expression in which translation and transcription are abstracted in a single process and transcriptional factors regulating gene expression. We treat explicitly the mechanisms of promoter repression and promoter activation through the representation of transcriptional factor bindings.

We follow the way opened by the influential work of Regev et al. [6], [13] on the ‘cells-as-computation’ abstraction. The approach suggests that the cell can be abstracted as a system of interacting computational entities. The computational entities are specified using the process algebra formalism. In Table 4.1 we present the classical mapping adapted to our approach.

Biology	Process Algebra
Entity	Process
Interaction capability	Enabled Action
Interaction	Multi-way Synchronization
Modification-Evolution	State Change

Table 4.1: The process algebra abstraction for systems biology adapted to PEPA.

In particular, as firstly suggested by Priami et al. [7], we apply a process algebra enriched with stochastic exponential distributed time durations in order to add time course aspects. Below we present in words the intuitive mapping of biological entities and activities to computational entities and activities.

- **Molecular populations:** a species population S is represented by a family of PEPA components $S(i)$. Only one of the individual processes $S(i)$ is active in the model at each time representing i molecules of that species.

- **Chemical reactions:** chemical reactions are abstracted as a synchronization between processes. The processes, that represents reagents and products species current populations, synchronize on an action called as the reaction name. When the action occurs, the processes change their internal state in order to represent a new species population, according to the reaction definition.
- **Operon structure:** an operon structure Op is represented as a process. The process can undertake an expression action in cooperation with all the processes representing the proteins for which it codes. The action causes a replenish of the protein populations. It also maintains, in its internal state, eventual information regarding activation or repression influences on its promoter region.
- **Transcriptional factors regulations:** a transcriptional factor regulation is represented by a synchronized action between the component representing the transcriptional factor population and the PEPA component representing the regulated protein coding region.

In Section 4.2 we firstly give the proper formal definition of the computational entities and activities in the form of PEPA components. Secondly we define how to merge these individual components in order to construct the complete PEPA component representing the final model.

4.2 Construction of the PEPA model

In this section we propose the automatic derivation rules for generating a PEPA model starting from a system expressed in the graphical notation. The input is a formal system definition $Sys=(\mathbf{Elements},\mathbf{Connections})$ expressed according to the constraints given in Section 3.4. The output is a PEPA model, given as a set of PEPA components definitions and the initial component configuration.

All the components we generate follow the structure specified in Figure 4.1. Each component has a name and a body, and the body is made of a list of PEPA processes, each element separated from the others by the choice operator.

During the automatic derivation, component names and component processes are constructed in different phases, thus we need to introduce supporting notations to keep track of their associations. We write $\underline{\text{list}}\langle Components \rangle.\underline{\text{add}}\langle Cname \rangle$ to add a new component with name $Cname$ in the special list $Components$ of the defined components.

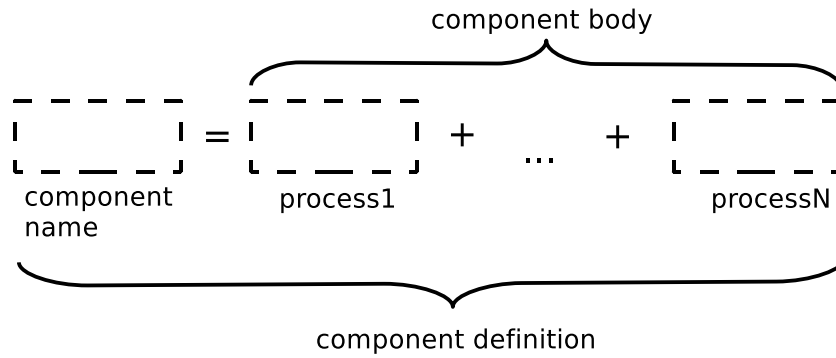


Figure 4.1: The syntactical structure of the automatically generated PEPA components.

We write $\underline{\text{list}}\langle Cname \rangle.\underline{\text{add}}\langle process \rangle$ when we want to add *process* to the process list of *Cname* component.

We need some meta-notation when writing the syntactical structure of the PEPA processes. They are constructed as concatenation of plain text and values taken from the objects in the **Elements** and **Connections** sets. The former are written in standard characters, the latter in bold. The lists' support commands are underlined.

4.2.1 Defining a species

For each object $S \in \mathbf{Elements}$ of type *TSpecies* we generate $(S.\mathbf{max} - S.\mathbf{min})$ process names. Given the identifier of the species $S.\mathbf{id}$, the processes are called $S.\mathbf{id}(i)$ with $S.\mathbf{min} \leq i \leq S.\mathbf{max}$. Each process represents a possible number of molecules of the species S , i.e. $S.\mathbf{id}(i)$ represents a population of i individuals.

$\underline{\text{list}}\langle Components \rangle.\underline{\text{add}}\langle S.\mathbf{id}(i) \rangle$ where $S.\mathbf{min} \leq i \leq S.\mathbf{max}$

In the PEPA model dynamics, only one of these processes is present at each time, representing the current species population.

4.2.2 Defining a reaction

For each object $R \in \mathbf{Elements}$ of type *TReaction* we generate a process definition with name $R.\mathbf{id}$.

$\underline{\text{list}}\langle Components \rangle.\underline{\text{add}}\langle R.\mathbf{id} \rangle;$

The process is able to synchronize on an action with name **R.id** and participate with a rate equal to the propensity coefficient **R.k**. After, the process returns to its original state.

$$\underline{\text{list}}\langle \mathbf{R.id} \rangle . \underline{\text{add}}\langle (\mathbf{R.id}, \mathbf{R.k}). \mathbf{R.id} \rangle ;$$

4.2.3 Defining a operon (or gene) expression

For each object **Op** \in **Connections** of type *TOperon* we generate a process name that represents the operon region state. We construct the name of the process adding the suffix *Op* at the promoter **P=Op.P** identifier.

$$\underline{\text{list}}\langle \text{Components} \rangle . \underline{\text{add}}\langle \mathbf{P.idOp} \rangle ;$$

At this step we generate also part of the body. We add an action called **P.idExp**, that represents the expression of the proteins for which the operon region codes. The exponential rate related to this is given by the attribute **P.e**. The process then returns to the initial state.

$$\underline{\text{list}}\langle \mathbf{P.idOp} \rangle . \underline{\text{add}}\langle (\mathbf{P.idExp}, \mathbf{P.e}). \mathbf{P.idOp} \rangle$$

For each species *S* there can exist objects **PcE** \in **Connections** of type *TProteinCod-Exp* where **PcE.S=S**. For object **PcE** there must exist one and only one object **Op** \in **Connections** of type *TOperon*. We add to the population processes of **S** an action by which its population is increased, caused by protein coding region expression.

$$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{Op.P.idExp}, 1). \mathbf{S.id}(i+1) \rangle, \text{ where } i < \mathbf{S.max}$$

4.2.4 Defining a reaction input

For each object **Ri** \in **Connections** of type *TReactionInput* we add body parts to the processes representing the species **S=Ri.S** that is reagent of the reaction **R=Ri.R**. We add in the body of each process level of species **S** an action that represents the participation as reagent to the reaction **R** and thus moves the species population to a state representing a lower population. The action is named after the reaction identifier **R.id**

and the exponential rate is the number of possible combinations of elements of species **S** for that level of population with regard to the stoichiometric coefficient. The stoichiometric coefficient is an attribute of the object **Rp=Rp.c**. The number of combinations for level *i* is called *Comb* and it is given by the formula:

$$Comb = \binom{i}{Rp.c}$$

we add a process definition to each component representing a level of **S** with enough substrate for enabling the reaction:

$$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{R.id}, Comb) . \mathbf{R.id}(i - \mathbf{Rp.c}) \rangle \text{ where } i \geq \mathbf{Rp.c}.$$

In the special case in which **S** is both input and output of the reaction see the Section 4.2.6.

4.2.5 Defining a reaction output

For each object **Ro** \in **Connections** of type *TReactionOutput* we add body parts to the processes representing the species **S=Ro.S** that is product of the reaction **R=Ro.R**. We add in the body of each process level of species **S** an action that represents the participation as product of the reaction **R** and thus moves the species population to a state representing an increased population. The action is named after the reaction identifier **R.id** and the exponential rate is 1 because products do not influence the rates of reactions governed by the mass action law. The stoichiometric coefficient is an attribute of the object **Rp=Ro.Rp**. We define the PEPA body definition:

$$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{R.id}, 1) . \mathbf{R.id}(i + \mathbf{Rp.c}) \rangle \text{ where } i \leq (\mathbf{S.max} - \mathbf{Rp.c}).$$

In the special case in which **S** is both input and output of the reaction see the Section 4.2.6.

4.2.6 Defining a reaction input-output

In the case of two objects $\mathbf{Ri} \in \mathbf{Connections}$ of type *TReactionInput* and $\mathbf{Ro} \in \mathbf{Connections}$ of type *TReactionOutput* having the same species and reaction attributes ($\mathbf{Ri.S}=\mathbf{Ro.S}$ and $\mathbf{Ri.R}=\mathbf{Ro.R}$) it is necessary to define a special conversion. This is because the species is both input and output of the reaction and so the increase and decrease in the species population need to be managed in a single PEPA action. The number of reagent combinations for level i is called *Comb* and it is given by the formula:

$$Comb = \binom{i}{\mathbf{Ri.Rp.c}}$$

We add a process definition to each component representing a level of \mathbf{S} with enough substrate for enabling the reaction but less than the species upper bound:

$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{R.id}, Comb) . \mathbf{R.id}(i - \mathbf{Ri.Rp.c} + \mathbf{Ro.R.c}) \rangle$, where $i \geq \mathbf{Ri.Rp.c}$ and $(i - \mathbf{Ri.Rp.c} + \mathbf{Ro.R.c}) \leq (\mathbf{S.max})$

4.2.7 Defining a promoter repression

For each object $\mathbf{Pr} \in \mathbf{Connections}$ of type *TPromoterRep* there exists only one object $\mathbf{Op} \in \mathbf{Connections}$ of type *TOperon* such that $\mathbf{Pr.P}=\mathbf{Op.P}$. We modify the body of the process representing the operon state that is called $\mathbf{Op.P.idOp}$. We add two sequential actions. The first is an action representing the binding of a molecule of Species $\mathbf{Op.S.id}$ to the promoter region; the second representing the unbinding of the same molecule. $\mathbf{P}=\mathbf{Op.P}$ is the promoter controlling the operon expression.

$\underline{\text{list}}\langle \mathbf{P.idOp} \rangle . \underline{\text{add}}\langle (\mathbf{P.id_bind_Pr.S.idRep}, \mathbf{Op.Rc.b}) . (\mathbf{P.id_unbind_Pr.S.idRep}, \mathbf{Op.Rc.r}) . \mathbf{P.idOp} \rangle$

Then we add a part to the body of the species component $\mathbf{S}=\mathbf{Op.S}$:

$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{P.id_bind_Pr.S.idRep}, i) . \mathbf{S.id}(i) \rangle$ where $i \neq 0$

4.2.8 Defining a promoter activation

For each object $\mathbf{Pa} \in \mathbf{Connections}$ of type $TPromoterAct$ there exists only one object $\mathbf{Op} \in \mathbf{Connections}$ of type $TOperon$ such that $\mathbf{Pa.P}=\mathbf{Op.P}$. We modify the body of the process representing the operon state that is called $\mathbf{Op.P.idOp}$. We add two sequential actions. The first is an action representing the binding of a molecule of Species $\mathbf{Op.S.id}$ to the promoter region; the second representing the unbinding of the same molecule associated with an additional expression of the operon region. $\mathbf{P}=\mathbf{Op.P}$ is the promoter controlling the operon expression.

$$\underline{\text{list}}\langle \mathbf{P.idOp} \rangle . \underline{\text{add}}\langle (\mathbf{P.id_bind_Pa.S.idAct}, \mathbf{Op.Rc.b}).(\mathbf{P.idExp}, \mathbf{Op.Rc.r}).\mathbf{P.idOp} \rangle$$

Then we add a part to the body of the species component $\mathbf{S}=\mathbf{Op.S}$:

$$\underline{\text{list}}\langle \mathbf{S.id}(i) \rangle . \underline{\text{add}}\langle (\mathbf{P.id_bind_Pa.S.idAct}, i).\mathbf{S.id}(i) \rangle \text{ where } i \neq 0$$

4.2.9 Defining the final components

When all the objects in the **Elements** and **Connections** sets have been derived, it is necessary to construct the PEPA components. For all the component names $Cname$ in $\underline{\text{list}}(\mathbf{Components})$ we construct the component:

$$Cname = process_1 + \dots + process_n$$

where $process_1, \dots, process_n$ are the entries in the $\underline{\text{list}}(Cname)$.

If the list is empty, the process is associated with the reserved action name *null* and defined having a simple circular behaviour:

$$Cname = (null, 1).Cname$$

4.2.10 Defining the root component

The previous instructions explained of how to generate the PEPA component definitions by deriving them from the graphical representation of the system. Here we give the rules to compose them into the initial configuration, also called the PEPA root com-

ponent. The root component is used as initial model definition from which the system dynamics are derived. It is therefore the first node of the derivation graph automatically generated by applying the PEPA semantics rules.

The set of the component definitions can be partitioned by separating the components into three mutually exclusive sets:

- **OP**, the set of components representing operon regions.
- **RP**, the set of components representing reactions.
- **SP**, the set of components representing the species populations.

For each species S_i we insert in the initial configuration the component definition $S_i.\text{id}(\mathbf{S}.\text{init})$ that represents the initial population species. We insert also the components $\{\mathbf{Op}_1, \dots, \mathbf{Op}_m\} \in \mathbf{OP}$, and the components $\{\mathbf{R}_1, \dots, \mathbf{R}_s\} \in \mathbf{RP}$. The components in the initial configuration I are connected with cooperation operators:

$$I = \mathbf{Op}_1 \underset{F()}{\bowtie} \dots \underset{F()}{\bowtie} \mathbf{Op}_m \underset{F()}{\bowtie} \mathbf{R}_1 \underset{F()}{\bowtie} \dots \underset{F()}{\bowtie} \mathbf{R}_s \underset{F()}{\bowtie} \mathbf{S}_1.\text{id}(\mathbf{S}.\text{init}) \underset{F()}{\bowtie} \dots \underset{F()}{\bowtie} \mathbf{S}_n(\mathbf{S}.\text{init})$$

$F()$ is the function that defines on which actions two cooperating components $A \underset{F()}{\bowtie} B$ can synchronize:

$$A \underset{F()}{\bowtie} B = A \underset{L}{\bowtie} B, \text{ where } L = (\mathcal{A}(A) \cap \mathcal{A}(B)) \setminus \{null\}$$

Where $\mathcal{A}(C)$ is a function, defined in Figure 4.2, that returns the set of the actions the PEPA component C is capable of. This set construction can also be stated as “components A and B cooperate only on the actions they are both capable of with the exception of the reserved action $null$ ”. It should be noticed that since the cooperation operator associates to the left, in the $A \underset{F()}{\bowtie} B$ construct the term A is usually not a single component but a cooperation of components.

4.3 Mechanistic interpretation of the stochastic process

Starting from the PEPA root component we can derive a continuous time Markov chain stochastic model by applying the PEPA semantics rules (details in Section 2.3). We therefore formally generate a state space set S , that represents all the possible configurations that the system can assume during time, and a transition rate matrix Q , that specifies how much time the system takes to transit from one state to another.

$$\begin{aligned}
\mathcal{A}((\alpha, r).P) &= \{\alpha\} \cup \mathcal{A}(P) \\
\mathcal{A}(P + Q) &= \mathcal{A}(P) \cup \mathcal{A}(Q) \\
\mathcal{A}(P \bowtie Q) &= \mathcal{A}(P) \cup \mathcal{A}(Q) \\
\mathcal{A}(P/\overset{\perp}{L}) &= \mathcal{A}(P)/L
\end{aligned}$$

Figure 4.2: Definition of the function \mathcal{A} . The function returns the set of the actions that a component is capable of.

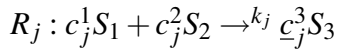
For construction, each state s_i (apart from intermediate configurations treated later) in the state space set ($s_i \in S$) can be written in the form:

$$s_i = \mathbf{Op}_1 \underset{\mathbb{F}()}{\bowtie} \dots \underset{\mathbb{F}()}{\bowtie} \mathbf{Op}_m \underset{\mathbb{F}()}{\bowtie} \mathbf{R}_1 \underset{\mathbb{F}()}{\bowtie} \dots \underset{\mathbb{F}()}{\bowtie} \mathbf{R}_s \underset{\mathbb{F}()}{\bowtie} \mathbf{S}_1(i_1) \underset{\mathbb{F}()}{\bowtie} \dots \underset{\mathbb{F}()}{\bowtie} \mathbf{S}_n(i_n)$$

where \mathbf{Op} are components representing operon regions, \mathbf{R} are components representing reactions and $\mathbf{S}(i)$ are components representing species (with population i). Three kind of transitions can happen from such a configuration: a reaction, a operon region expression, a transcriptional regulation process. In the next subsection, we formally derive for each possible transition the resulting arrival system state and we discuss the meaning of the associated exponential rate.

4.3.1 Reactions

We assume that in state s_i a reaction R_j has enough molecules of the substrates to occur. For simplicity we assume it to be between products S_1 and S_2 in order to make product S_3 (the derivation for higher orders of reagent and products comes straightforwardly from this simplified case). Given the reaction in the chemical form:



we know that for construction the current state s_i is in the form:

$$s_i = \dots \underset{\mathbb{F}()}{\bowtie} \mathbf{S}_1(i_1) \underset{\mathbb{F}()}{\bowtie} \mathbf{S}_2(i_2) \underset{\mathbb{F}()}{\bowtie} \mathbf{S}_3(i_3) \underset{\mathbb{F}()}{\bowtie} \mathbf{R}_j$$

and the components representing the species and the reaction that has to occur are in the form:

$$\mathbf{S}_1(i_1) = \dots + (R_j, r_1) \cdot \mathbf{S}_1(i_1 - c_j^1), \text{ where } r_1 = \binom{i_1}{c_j^1}$$

$$\mathbf{S}_2(i_2) = \dots + (R_j, r_2) \cdot \mathbf{S}_2(i_2 - c_j^2), \text{ where } r_2 = \binom{i_2}{c_j^2}$$

$$\mathbf{S}_3(i_3) = \dots + (R_j, 1) \cdot \mathbf{S}_3(i_3 + c_j^3)$$

$$\mathbf{R}_j = (R_j, k_j) \cdot \mathbf{R}_j$$

and they can synchronize on action R_j since $\{R_j\} \subseteq \mathcal{A}(\mathbf{S}_1) \cap \mathcal{A}(\mathbf{S}_2) \cap \mathcal{A}(\mathbf{S}_3) \cap \mathcal{A}(\mathbf{R}_j)$. The four components can therefore join in a multi-way cooperation that generates a transition to the state s_j :

$$s_j = \dots \underset{\mathbb{F}()}{\boxtimes} \mathbf{S}_1(i_1 - c_j^1) \underset{\mathbb{F}()}{\boxtimes} \mathbf{S}_2(i_2 - c_j^2) \underset{\mathbb{F}()}{\boxtimes} \mathbf{S}_3(i_3 + c_j^3) \underset{\mathbb{F}()}{\boxtimes} \mathbf{R}_j$$

and according to cooperation semantics rule the rate of the transition from s_i to s_j is $q_{ij} = r_1 * r_2 * k_j$. This means that we associate to each reaction to occur a probability (and duration) that is distributed according to an exponential distribution with rate proportional to the combinations of the involved reagents. Moreover the arrival state represents a molecular population rigorously defined from the stoichiometric coefficients of the occurred reaction.

This result is of great interest when we consider models of elementary reactions as in the Gillespie Algorithm (see Section 2.6) scenario. Under the same fundamental assumptions, if we set the constant k_j as the stochastic constants for the reaction, the CTMC underlying the model happens to be exactly the one that identifies the evolution of the system's *Master equation*. This can be easily seen when comparing the transition matrix Q entries with the propensity coefficients calculated at each run in *step 2* by the *Gillespie Algorithm* (details in Section 2.6). From the theoretical point of view, this assures us that our model is sound with the rigorous and exact derivation of the *Master equation*. From the practical point of view, this says that if we perform a probabilistic random walk on the CTMC, we are basically running an algorithm equivalent in calculation to the Gillespie one.

4.3.2 Operon (or gene) expressions

We assume that in state s_i the expression of the operon region Op_1 can occur, meaning that it is enabled and the species for which it codes are not at the maximal level. We assume for simplicity that the operon codes for only on protein S_1 . Therefore the current state s_i is in the form:

$$s_i = \dots \underset{F()}{\boxtimes} S_1(i_1) \underset{F()}{\boxtimes} Op_1$$

and the components representing the species and the gene expression that has to occur are in the form:

$$S_1(i_1) = \dots + (Op_1Exp, 1).S_1(i_1 + 1)$$

$$Op_1 = \dots + (Op_1Exp, e).Op_1$$

and they can synchronize on action Op_1Exp since $\{Op_1Exp\} \subseteq \mathcal{A}(S_1) \cap \mathcal{A}(Op_1)$. The two components can therefore join in a cooperation that generates a transition to the state s_j :

$$s_j = \dots \underset{F()}{\boxtimes} S_1(i_1 + 1) \underset{F()}{\boxtimes} Op_1$$

and according to the cooperation semantics rule the rate of the transition from s_i to s_j is $q_{ij} = e$. Therefore the underlying assumption of this approach is that a coding region expresses a protein with a duration following an exponential distribution with parameter e . To the best of our knowledge, there is no scientific evidence of such a behaviour of the actual biology entity. However the exponential assumption is mandatory for the construction of a CTMC stochastic process and it is usually accepted when no better characterization is known. A similar approach for modelling gene expression has been firstly proposed by Cardelli et al. [16] in a work limited only to gene networks and using π -calculus (see Section 2.2).

4.3.3 Transcriptional regulation processes

A transcriptional regulation process can be a promoter activation or repression. Firstly we consider the repression scenario. S_1 is a species defined as transcriptional regulator

factor for the operon region \mathbf{Op}_1 . Therefore current state s_i is in the form:

$$s_i = \dots \underset{\mathbb{F}()}{\boxtimes} \mathbf{S}_1(i_1) \underset{\mathbb{F}()}{\boxtimes} \mathbf{Op}_1$$

and the components representing the species and the operon region are in the form:

$$\mathbf{S}_1(i_1) = \dots + (\mathbf{P_bind_SRep}, i) . \mathbf{S}_1(i_1)$$

$$\mathbf{Op}_1 = \dots + (\mathbf{P_bind_SRep}, b) . (\mathbf{P_unbind_SRep}, r) . \mathbf{Op}_1$$

and they can synchronize on action $\mathbf{P_bind_SRep}$ since $\{\mathbf{P_bind_SRep}\} \subseteq \mathcal{A}(\mathbf{S}_1) \cap \mathcal{A}(\mathbf{Op}_1)$. The two components can therefore join in a cooperation that generates a transition to the state s_j :

$$s_j = \dots \underset{\mathbb{F}()}{\boxtimes} \mathbf{S}_1(i_1) \underset{\mathbb{F}()}{\boxtimes} (\mathbf{P_unbind_SRep}, r) . \mathbf{Op}_1$$

and according to the cooperation semantics rule the rate of the transition from s_i to s_j is $q_{ij} = b * i$. Therefore the underlying assumption of this approach is that each transcriptional factor molecule in population S has a independent exponential instantaneous probability b of binding to the free operon region. When the binding occurs, the operon regions enter in an inhibited state in which expression is disabled. This state lasts until an action $\mathbf{P_unbind_SRep}$ occurs. That action represents the unbinding of the molecule and has exponential rate r .

The activation scenario follows the same structure except that the subsequent action to the binding is an operon expression, thus replenishing the species for which it codes, in addition to the constitutive expression.

A similar approach for modelling gene repression and activation has been firstly proposed by Cardelli et al. [16] in a work limited only to gene networks and using π -calculus (see Section 2.2).

4.4 Translation from PEPA to PRISM

In this section we propose an automatic translation from a model developed in PEPA language to a model in PRISM language. The conversion is merely syntactical such that the probabilistic process defined is the same continuous time Markov chain. PRISM

is a probabilistic symbolic model checker that provides support for the analysis of probabilistic models, such as continuous time Markov chains (see Section 2.4). There are several reasons for which we choice to convert the models to PRISM. Firstly, as explained in Section 2.3, the ad hoc analysis tools developed for the PEPA language (i.e. the PEPA Workbench) are not suitable for our purpose. They implement the original PEPA semantics (Figure 2.3) and are not easily extendible to the slightly modified one that we defined (Figure 2.4). Secondly, PRISM permits state-of-the-art CTMCs analysis features such as *Monte-Carlo simulation*, *probabilistic model checking*, *transient* and *steady state* analysis, *manual debugging* and offers a *graphical user interface*. Moreover, PRISM already offers the function to compile models written in PEPA to its format. Again only the original PEPA semantics is implemented, but the theoretical mapping between the two languages is established. Therefore we do not need to prove the existence of such a mapping, and we simply present in Table 4.2 some pseudo-code translation rules. Informally, each PEPA *component* is translated into a PRISM *module* and each PEPA *action* into a PRISM *command*. The components that represent reactions are translated as actions in the module *Rates*. PRISM *modules* run in parallel and act like cooperative PEPA *components*. PRISM *variables* are used to replicate the sequential structure of PEPA processes.

Entity	in PEPA	in PRISM
Molecules definitions for protein A	$A(\min A)$... $A(\max A)$	module A levelA[$\min A$.. $\max A$] init initA; endmodule
Molecules population increasing of j molecules.	$A(i) = \dots + (act, rate).A(i + j)$	<i>in module A:</i> [act] (levelA+j ≤ $\max A$) : rate → levelA' = levelA + j;
Molecules population decreasing of j molecules.	$A(i) = \dots + (act, rate).A(i - j)$	<i>in module A:</i> [act] (levelA - j ≥ $\min A$) : rate → levelA' = levelA - j;
Protein A repressor for operon region Op.	$A(i) = \dots + (bind_rep, rate).A(i)$	<i>in module A:</i> [bind_rep] (levelA > 0) : rate → levelA' = levelA;
Protein A activator for operon region Op.	$A(i) = \dots + (bind_act, rate).A(i)$	<i>in module A:</i> [bind_act] (levelA > 0) : rate → levelA' = levelA;
Expression of operon region Op.	$Op = \dots + (OpExp, rate).Op$	module Op [OpExp] (check=0) : rate → true; endmodule
Activation of operon region Op by protein A.	$Op = \dots + (bind_act, rate). (OpExp, rate2).Op$	<i>in module Op:</i> [bind_act] (check=0) : rate → check=1; [OpExp] (check=1) : rate2 → check=0;
Repression of operon region Op by protein A.	$Op = \dots + (bind_rep, rate). (unbind_rep, rate2).Op$	<i>in module Op:</i> [bind_rep] (check=0) : rate → check=1; [unbind_rep] (check=1) : rate2 → check=0;
Reaction R.	$R = (act, rate).R$	<i>in module Rates</i> [act] (true) : rate → true; endmodule

Table 4.2: Rules for the conversion from PEPA syntactical structures to PRISM code.

Chapter 5

Modelling and analysis in Synthetic Biology

The current *Synthetic Biology* ability to handle organisms is arguably focused at the molecular level of the cell. Therefore, a valid modelling framework must offer primitives able to easily represent the molecular fundamental processes that regulate the living organisms' building-block internal and external behaviours. These processes are inherently complex because they are the results of complicated biological networks, in which functional and multifunctional entities interact in a selectively and often non-linear fashion [5]. In this complicated scenario, human understanding tries to categorise networks according to patterns and functionalities: *regulatory gene networks*, *metabolic networks*, *signalling pathway networks*, *membrane networks* and so on. On the top of current biological knowledge, in the discipline of Systems Biology the paradigms of the individual mechanisms underlying these networks' behaviours have been recently identified. In Section 5.1, Section 5.2 and Section 5.3 we aim to show that the stochastic process algebra approach is suited to representing these fundamental mechanisms. In the first we focus on *biochemical reaction networks*, in the second on *gene regulatory networks* and in the last on *membrane and transport networks*. In Section 5.4 we present the model of a real synthetic system that was realized in the context of the iGEM 2006 competition.

For each example we report the automatically generated PEPA model and time-course plots of the system behaviour calculated using stochastic simulations. When meaningful, we analyse the models using probabilistic model checking techniques. The PRISM codes can be found in the Appendix.

5.1 Common patterns in biochemical reactions' networks

Some living organisms main processes, such as metabolism and signalling response, rely on a constant dynamical flux of material and information propagated by highly interconnected networks of chemical reactions [28]. In these contexts, the term reaction identifies processes with possibly very different physical mechanisms.

Metabolic networks consist mainly of chemical transformation of one type of molecules into another type: *catabolic reactions* are breakdowns of complex compounds to obtain energy and raw material, *anabolic reactions* are constructions of complexes of smaller molecules for cellular functioning needs. In both, the reactions are usually catalyzed by enzymes that act as central regulatory nodes in the reagents-to-products fluxes. Enzymatic reactions can be activated or inactivated by particular molecules called effectors.

Reactions in *Signalling pathway networks* involve the same type of mechanisms as in *metabolic networks* but the substrate modifications are usually achieved by activation-inactivation through phosphotransfer systems, resulting in phosphorylations, methylations or acetylations.

In the next subsections we show how the principal type of reactions can be modelled in the *Synthetic Biology* framework we are proposing.

5.1.1 Proteins' Degradation

Proteins are polypeptide chains made of amino acids that are essential parts of the organisms and participate in every process within the cell. The concentrations at which they are present in the cytosol or in the cell's other regions is therefore a fundamental aspect for understanding biomolecular mechanisms. The compounds' concentrations in time are the result of synthesis and degradation processes. At the steady-state level the concentration is maintained by the balance between synthesis and degradation rates. In nature, the degradation process is the result of a series of complicated interactions with enzymatic pathways. In modelling, the process is usually abstracted as a single reaction representing the disgregation of the polypeptide chain. Each single protein is thought to have an independent probability of undertaking degradation, as in Figure 5.1.

Here we discuss the simple example of protein A present in the cell in the number of a one hundred molecules. Each individual has an exponentially distributed probability with parameter $kDeg$ of being degraded. The automatically derived process algebra

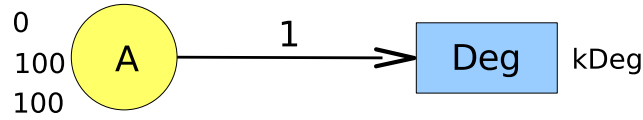


Figure 5.1: Graphical representation of the Protein's Degradation pattern.

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(null, 1).A(0)$	
$A(i) =$	$(Deg, i).A(i - 1)$	$1 \leq i \leq 100$
$Deg =$	$(Deg, kDeg).Deg$	

PEPA system definition
$A(100) \xrightarrow{\{Deg\}} Deg$

Number of states: 101
Number of transitions: 101

Table 5.1: PEPA model definition and information about the associated CTMC for the Protein's Degradation pattern.

model is composed of one hundred and two simple processes that could cooperate on the degradation action Deg (PEPA code in Table 5.1, PRISM code in Appendix A.0.1). Of the one hundred processes representing A population, only one is active at each time. An occurrence of the action leads to the decrease of A 's population of one unit.

The resulting stochastic process is a *continuous time Markov chain* with few states and transitions. Each state represents a possible population of A (from 0 to 100) and from each state there is a degradation transition. Intuitively, the velocity of degradation of the protein population is faster when the population is bigger and slower when it reaches few elements. In Figure 5.2 there are stochastic simulations that show the behavioural change when the degradation probabilistic rate constant is changed. The first plot presents the time course of a single simulation run where the latter presents the average behaviour of five hundred simulation runs.

Using model checking techniques is possible to calculate exact probabilities of the behaviour of the system, instead of relying on approximate information given by averages of simulation runs. For example we may ask “*what is the probability of there being i molecules of A at time T ?*”. The corresponding CSL query is given by the formula $P_{\bowtie}[\text{true}\mathbf{U}^{[t,t]}(A = i)]$. Figure 5.3 reports the resulting distribution of A popu-

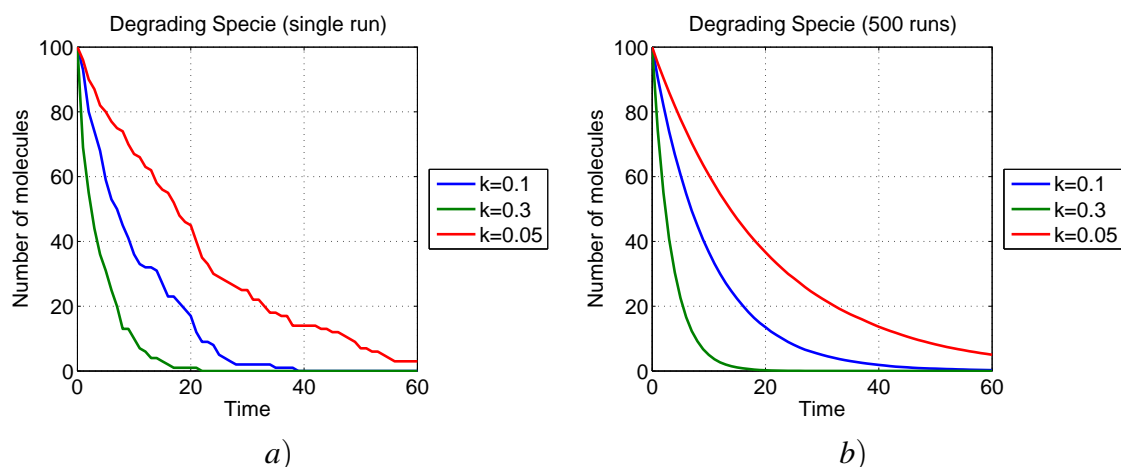


Figure 5.2: Stochastic simulations for the Protein's Degradation pattern with $\text{InitA}=100$ and different values for k_{Deg} . a) Single simulation run. b) Average of 500 simulation runs.

lation with i ranging from 0 to 100 and time ranging from 0 to 100 with a discretized step of 1 unit.

5.1.2 Irreversible and Reversible Reactions

In living cells, biochemical reactions are fundamental processes that help sustain life and allow cells functioning. The term refers to the broad range of chemical reactions which take place in all living organisms. Examples of mechanisms for which biochemical reactions occur are oxidation, reduction, movement of functional groups, bond-breaking and bond-forming reactions. To name only a few of thousands, in the organisms they are responsible of processes such as conversion of food into energy, respiration and nerve impulse reaction. Reactions happen at the molecular level as collection of *elementary processes* (also called *elementary steps* or *elementary reactions*) that contribute to the overall reaction occurrence. The sequence of elementary processes explain to us the actual physical flow of the reaction.

Since it is very difficult to identify the actual elementary processes of a reaction, proposal of *mechanisms* are usually derived from experimental observation. A *mechanism* is a rationalization of a chemical reaction. In our modelling scenario we suppose to have *mechanisms* for all the reactions we seek to represent. Therefore we represent them as networks of elementary reactions in which substrates interacts in order to form products. Under this assumption our modelling framework generates time rates for the reaction according to the law of *mass action*. An irreversible reaction is a chemical

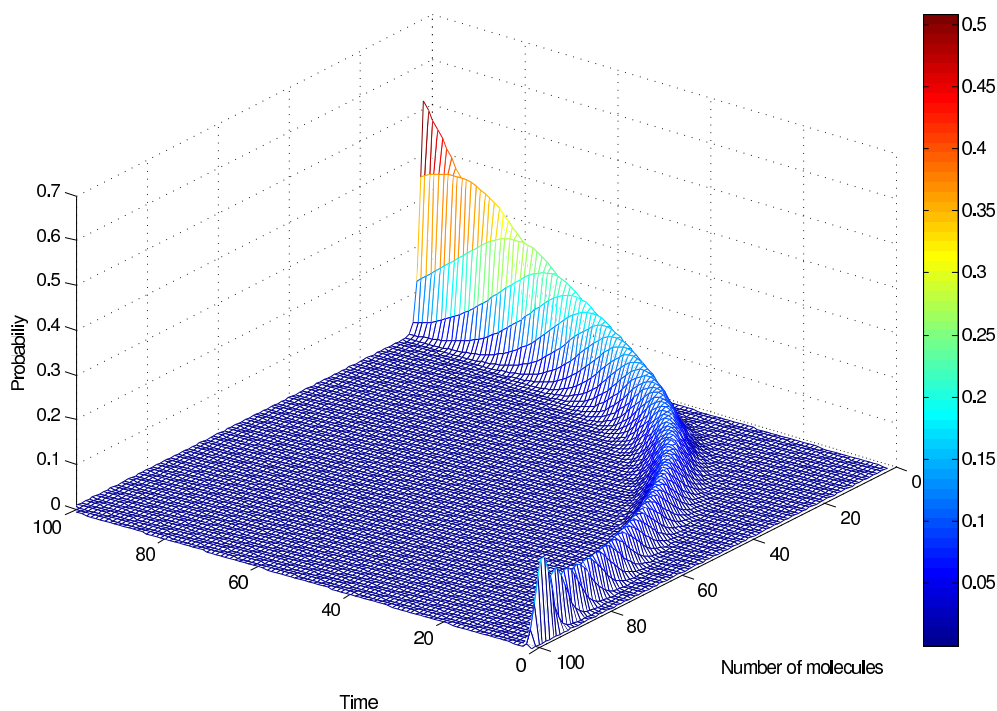


Figure 5.3: Plot representing the probabilistic distribution of A protein population in time in the Protein's Degradation pattern (initA=100, kDeg=0.8). The probability of having i molecules in time t has been calculated using model checking techniques. Time has been discretized with a unitary step from 0 to 100.

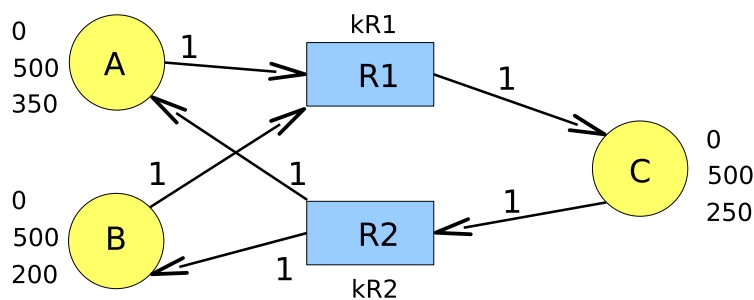


Figure 5.4: Graphical representation of the Reversible Reaction pattern.

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(R2, 1).A(1)$	
$A(i) =$	$(R1, i).A(i-1) + (R2, 1).A(i+1)$	$1 \leq i \leq 499$
$A(500) =$	$(R1, i).A(499)$	
$B(0) =$	$(R2, 1).B(1)$	
$B(i) =$	$(R1, i).B(i-1) + (R2, 1).B(i+1)$	$1 \leq i \leq 499$
$B(500) =$	$(R1, 500).B(499)$	
$C(0) =$	$(R1, 1).C(1)$	
$C(i) =$	$(R2, i).C(i-1) + (R1, 1).C(i+1)$	$1 \leq i \leq 499$
$C(500) =$	$(R2, i).C(499)$	
$R1 =$	$(R1, kR1).R1$	
$R2 =$	$(R2, kR2).R2$	

PEPA system definition
$(A(350) \bowtie_{\{R1, R2\}} B(200) \bowtie_{\{R1, R2\}} C(250) \bowtie_{\{R1\}} R1) \bowtie_{\{R2\}} R2$

Number of states: 351
Number of transitions: 700

Table 5.2: PEPA model definition and information about the associated CTMC for the Reversible Reaction pattern.

reaction that takes place in only one direction and thus proceeds finally to completion. At the contrary a reversible reaction can proceed from substrate to products and vice versa. In Figure 5.4 we present an example considering species A and B that can perform reaction $R1$ to form species C (for example as complexation) and considering species C that can perform reaction $R2$ to form the species A and B (for examples as decomplexation).

The PEPA model in Table 5.2 represents the reversible reaction R modelled as combination of reaction $R1$ and $R2$. Figure 5.5 shows the simulated behaviour of the reaction when only the forward reaction $R1$ is considered active and compares it when also the backward reaction $R2$ is considered. As expected, in the first case the reaction proceed until consuming of substrates A and B . In the latter case, since the rate of reaction $R2$ is faster, the steady-state is assested at the concentrations' levels where forward and backward reaction rates are at the equilibrium (PRISM code in Appendix A.0.2).

It is possible to use CSL logic in order to investigate the probability of a protein population to be in a certain numeric range. For example we may ask ‘*what is the*

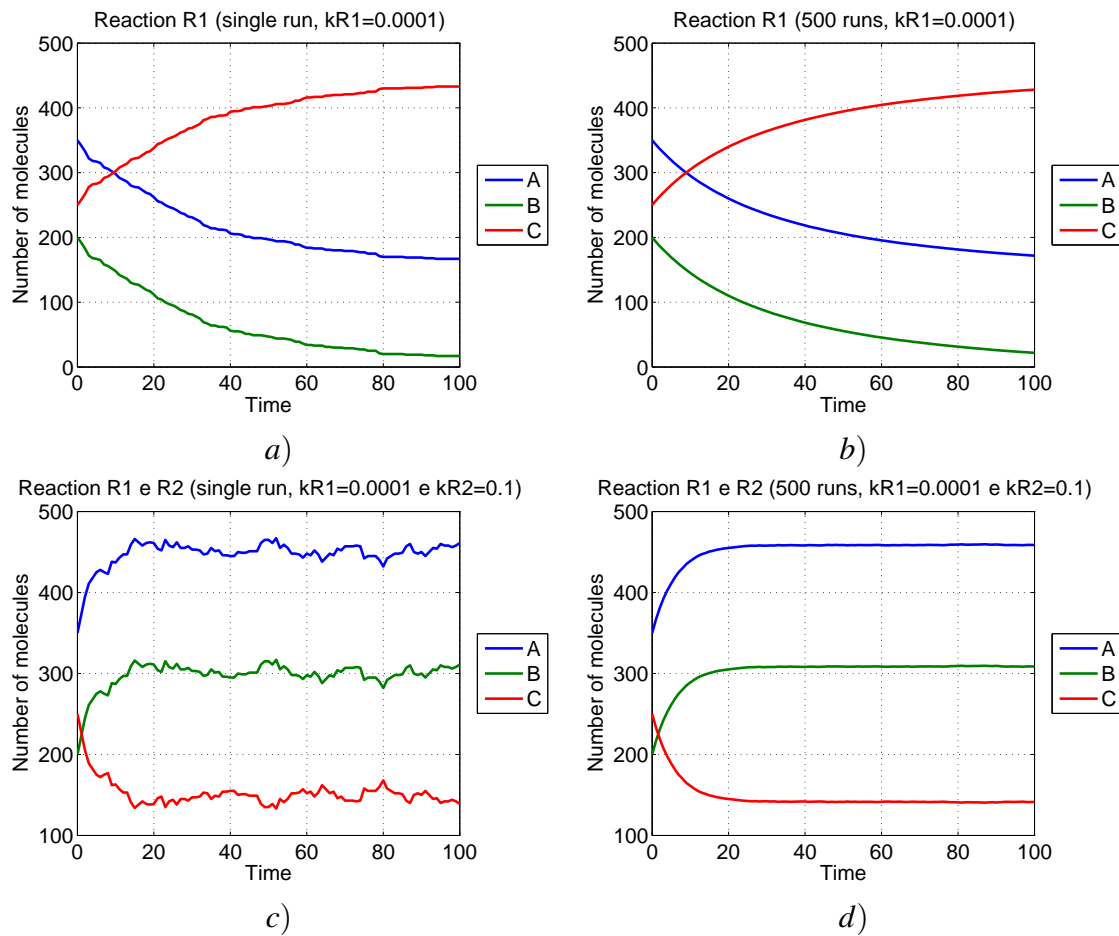


Figure 5.5: Stochastic simulations for the Irreversible and Reversible Reaction pattern. a) Single simulation run of the irreversible reaction R1 (R2 is not considered). b) Average of 500 simulation runs of the irreversible reaction R1 (R2 is disabled). c) Single simulation run of the reversible reaction R (R1 and R2 considered). d) Average of 500 simulation runs of the reversible reaction R (R1 and R2 considered).

probability that population of protein X will be in between a lower bound lb and an upper bound ub at time T?". The question can be translated as the probabilistic logic formula $P_{\times} [true \mathbf{U}^{[t,t]} (lb < X < ub)]$. Figure 5.6 shows the probabilities of protein A,B and C populations being respectively in between 430 and 470, 290 and 310, 145 and 155 molecules, in the time interval from 0 to 100 (time discretized with unitary step).

5.1.3 Enzymatic Reactions

Enzymes are large protein molecules that act as biological catalysts, occurring in reactions as chemical accelerator without being consumed. The enzymes' concentrations and activities inside the cell are fundamental aspects for the velocity and regulation

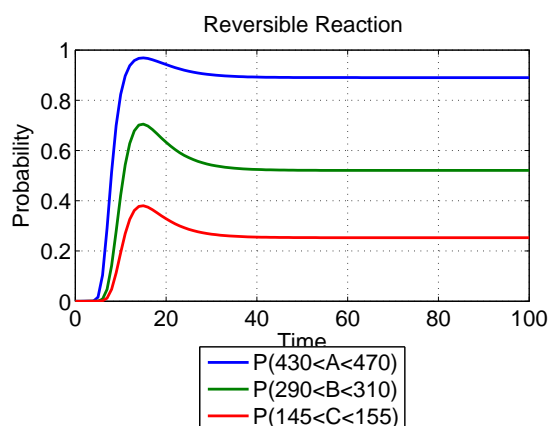


Figure 5.6: Plot representing the probabilities of A,B and C protein populations to be in a particular range in time period 0 to 100 (time discretized with unitary step).

of chemical transformation fluxes. Enzymes activity is specific for a certain set of reactions and chemical substrates and depends in efficiency on external factors such as pH and temperature. The common *mechanism rationalization* for enzymatic reactions proposes them as the combination of two elementary reactions: the reversible binding of a substrate molecule to an enzyme molecule and the irreversible transformation of such complex into a product individual and the enzyme itself. In our modelling framework we represent such reactions as in Figure 5.7. The automatic generated PEPA model (in Table 5.3) is composed of the processes representing the enzymes molecules E , the reagents molecules A , the complex $A-E$ and the product P . Reactions $R1$ and $R2$ represent the reversible reaction between E and A , reaction $R3$ represent the irreversible reaction between $A-E$ and P . The stochastic simulations graphs (Figure 5.8) help in understanding the time-course of the reaction. The binding rate of enzyme and substrate is faster of several degrees than the production rate, thus in the system the formation of complex $A-E$ build up early. The production of P is thus regulated by the available maximal amount of E and leads to a complete transformation of A in P with a linear behaviour. The PRISM code is listed in Appendix A.0.3.

In addition to their effect of reaction fastener, enzymes are also involved in metabolic regulation in various way. Effectors, that are proteins or other molecules, can bind with different species involved in the enzymatic reaction causing a reaction inhibition. The mechanisms of inhibition can be several, for example the irreversible binding of effectors to enzymes or to the enzyme-substrate complex. These inhibition regulations can be modelled in our framework by adding to the template representation the effector species and its interaction with the regulated species.

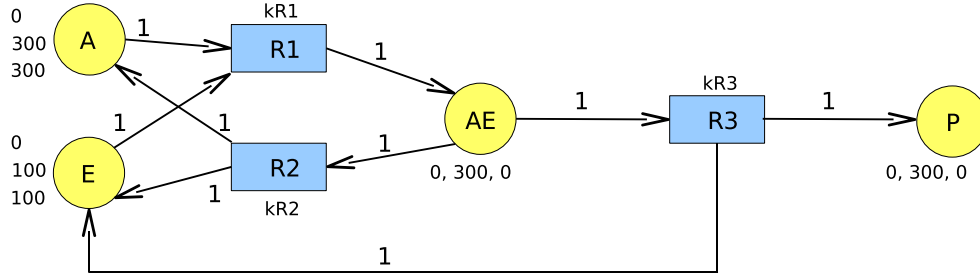


Figure 5.7: Graphical representation of the Enzymatic Reaction pattern.

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(R2, 1).A(1)$	
$A(i) =$	$(R1, i).A(i-1) + (R2, 1).A(i+1)$	$0 < i < 300$
$A(300) =$	$(R1, 300).A(299)$	
$E(0) =$	$(R2, 1).E(1) + (R3, 1).E(1)$	
$E(i) =$	$(R1, i).E(i-1) + (R2, 1).E(i+1) + (R3, 1).E(i+1)$	$0 < i < 100$
$E(100) =$	$(R1, 100).E(99)$	
$AE(0) =$	$(R1, 1).AE(i+1)$	
$AE(i) =$	$(R2, i).AE(i-1) + (R3, i).AE(i-1) + (R1, 1).AE(i+1)$	$0 < i < 300$
$AE(300) =$	$(R2, 300).AE(299) + (R3, 300).AE(299)$	
$P(i) =$	$(R3, 1).P(i+1)$	$0 \leq i < 300$
$P(300) =$	$(null, 1).P(300)$	
$R1 =$	$(R1, kR1).R1$	
$R2 =$	$(R2, kR2).R2$	
$R3 =$	$(R3, kR3).R3$	

PEPA system definition
$((A(300) \bowtie_{\{R2, R1\}} E(100) \bowtie_{\{R1, R2, R3\}} AE(0) \bowtie_{\{R3\}} P(0) \bowtie_{\{R1\}} R1) \bowtie_{\{R2\}} R2) \bowtie_{\{R3\}} R3)$

Number of states: 25351
Number of transitions: 75151

Table 5.3: PEPA model definition and information about the associated CTMC for the Enzymatic Reaction pattern.

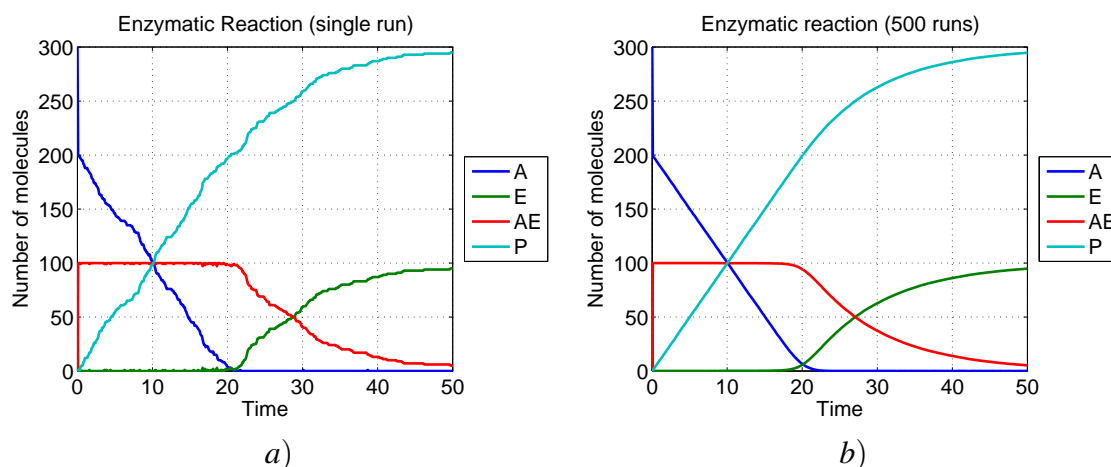


Figure 5.8: Stochastic simulations for the Enzymatic Reaction pattern with $kR1=1$, $kR2=0.01$, $kR3=0.1$. a) Single simulation run. b) Average of 500 simulation runs.

5.2 Common patterns in gene regulatory networks

In living organisms, the genome is involved in controlling fundamental cellular processes such as response to external stimulus, the replication of genetic informations before the division, the regulation of metabolic fluxes and cell's differentiation [8]. *Gene regulatory networks* (GRNs) are therefore of immense importance for understanding and controlling the cell's behaviour. The DNA structures made of nucleotide sequence regions contain the informations used to synthesize proteins, which in turns might be transcription regulation factors for other proteins synthesis or might participate in *metabolic* and *signalling networks*. The synthesis of a protein is a complex process compose of several steps that start with the transcription of the DNA nucleotide sequence in RNA, thus in turn is transformed in mRNA and finally in the protein. In between each step, various form of regulations occur in order to control the final expression efficiency for the particular protein, therefore regulating the functions in which it is involved. For *Synthetic Biology*, the study of gene expression regulation caused by *gene regulatory network* interactions is of fundamental importance, because the extent at which we can modify organisms is at the moment limited to introduction of *synthetic gene sequences* (see Section 2.1).

The data obtained from the sequencing projects combined with experimental and system oriented analysis permits the characterization of DNA sequencing with functional abilities. The *Registry of Standard Biological Parts* is a library of functional parts such as promoters, protein coding regions and terminators. In the next subsections we

present our modelling approach when functional parts are composed to obtain *gene regions expression* and *transcriptional factor gene regulations*. In the last subsection we propose a specific representation for the *transcription* and *translation* phases.

5.2.1 Gene Expression

A gene is a DNA sequence composed of functional and non functional parts. Broadly speaking, a gene usually contains a promoter region, which controls the activity of a gene, and coding and non-coding sequences. The coding sequences determine the product of gene expression while the non-coding sequences might regulate other aspects of expression. Using the graphical notation it is possible to represent a operon structure or a gene structure. The operon structure, usually found in prokaryotes, consist of a promoter region followed by several regions that code for different proteins. In the common definition of gene, the protein coding regions are limited to one.

In Figure 5.9 we represent a gene composed of a promoter region P that controls the expression of the protein coding region PCA . PCA codes for protein A . The degradation of the protein A is also considered as reaction Deg . In this representation the promoter does not have any transcriptional factors interactions therefore it is supposed to be constitutively expressing the downstream region with an exponentially distributed rate eP . The automatic generated PEPA model (in Table 5.4, PRISM code in Appendix A.0.4) is composed of three processes representing the gene region, the degradation reaction and the protein population. Stochastic simulations of the systems can be seen in Figure 5.10. Leaving the degradation stochastic constant rate unchanged, we vary the expression stochastic constant in order to investigate the population size at the steady state and during the building up phase.

Using model checking formula $P_{\geq \alpha}[\text{true}\mathbf{U}^{[t,t]}(A = i)]$ is possible to investigate “*what is the probability of there being i molecules of A at time T* ”. Figure 5.11 shows the resulting distribution of A population with i ranging from 0 to 100 and time ranging from 0 to 100 with a discretized step of 1 unit.

5.2.2 Gene Activation and Repression

The mechanisms of gene regulation operates at several levels. One fundamental is the binding of regulatory proteins, called *transcriptional factors*, to specific sites on the promoter region in order to change the transcriptional rate of the downstream coding regions. An *activator* is a transcriptional factor that causes an speed up of the rate

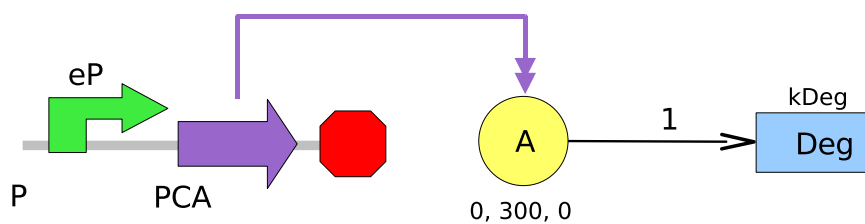


Figure 5.9: Graphical representation of the Gene Expression pattern.

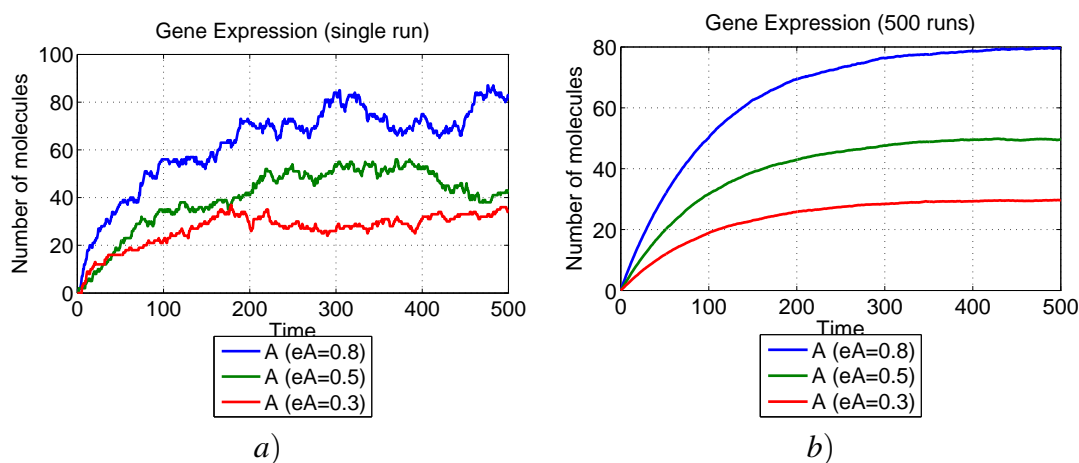
PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(PExp, 1).A(1)$	$1 \leq i \leq 299$
$A(i) =$	$(Deg, i).A(i-1) + (PExp, 1).A(i+1)$	
$A(300) =$	$(Deg, 300).A(299)$	
$Deg =$	$(Deg, kDeg).Deg$	
$POp =$	$(PExp, eP).POp$	

PEPA system definition

$$(A(0) \underset{\{PExp\}}{\bowtie} POp) \underset{\{Deg\}}{\bowtie} Deg$$

Number of states: 301**Number of transitions: 600**

Table 5.4: PEPA model definition and information about the associated CTMC for the Gene Expression pattern.

Figure 5.10: Stochastic simulations for the Gene Expression pattern with $kDeg=0.01$ and different values for eP . a) Single simulation run b) Average of 500 simulation runs.

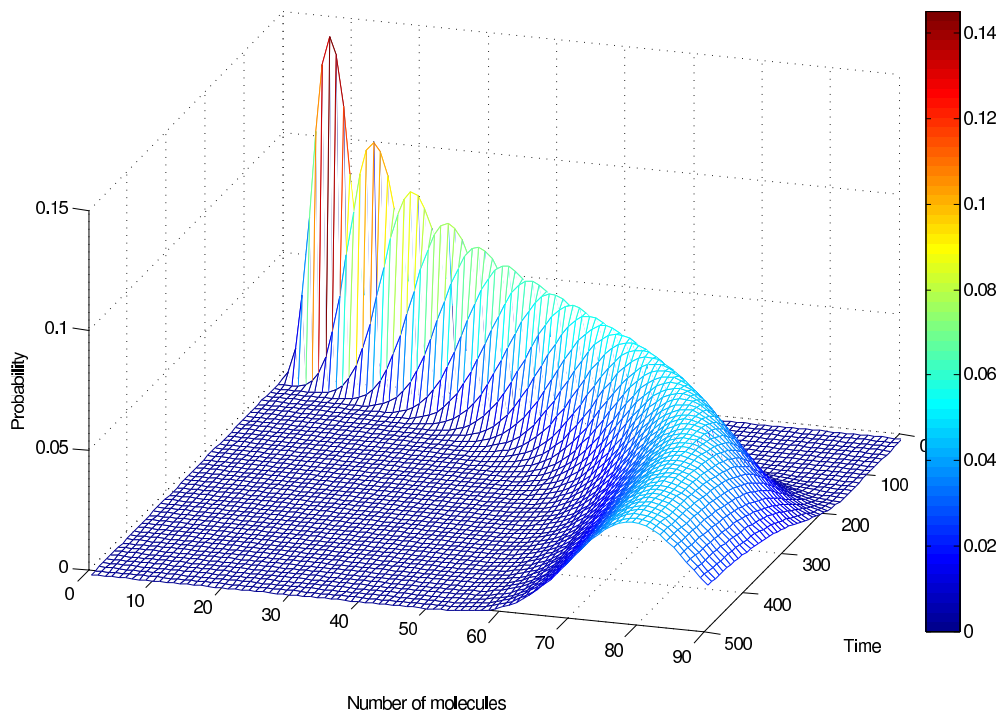


Figure 5.11: Plot representing the probabilistic distribution of A protein population in time in the Gene Expression pattern ($\text{initA}=0$, $\text{kDeg}=0.01$, $\text{kP}=0.8$). The probability of having i molecules in time t has been calculated using model checking techniques. Time has been discretized with a unitary step from 0 to 100

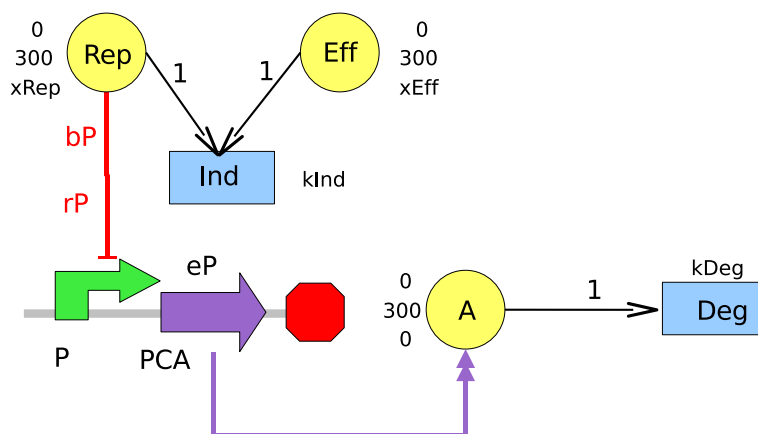


Figure 5.12: Graphical representation of the Gene Repression and Inducible System pattern.

by enhancing the attraction of RNA polymerase (*positive control*). At the contrary, a *repressor* is a transcriptional factor that causes a slow down of the rate by disturbing the bind of RNA polymerase on the promoter site (*negative control*). In general prokaryotic cells are controlled by *negative control systems*, meaning that prokaryotic genes are expressing by default and RNA polymerase does not require any factor to initiate transcription. Eukaryotic organisms generally are governed by *positive control systems*, meaning that eukaryotic genes are not expressing by default and RNA polymerase requires general and specific transcription factors to initiate transcription.

The actual biological mechanisms can be very different but such categorization is useful for description and analysis purposes.

The effect of the transcriptional factors population on the transcription rate is influenced by characteristics such as the affinity with the binding region site, the concentration of the factors, the presence of co-factors and so on. In the context of *Synthetic Biology*, activator and repressor effects play a major role in designing *on* and *off* switches of protein region expression. At the time of this writing, in the *Registry of Standard Biological Parts* are present 22 *coding regions* for *repressors* and *activators proteins* together with more than 50 *promoter sequences* than might be composed in order to obtain complex regulatory behaviours. Examples for the repressor category are *Lac* and *Tet* while *Lamba CI* is an exponent for the activation category.

In Figure 5.12 it is presented a simply regulatory scenario with a constitutively *on* promoter *P* that can be repressed by protein *Rep* (*xEff* initial population is set to zero) and in Table 5.5 the relative PEPA code (PRISM code in Appendix A.0.6). In Figure 5.14 are proposed stochastic simulations of the system with different concentrations

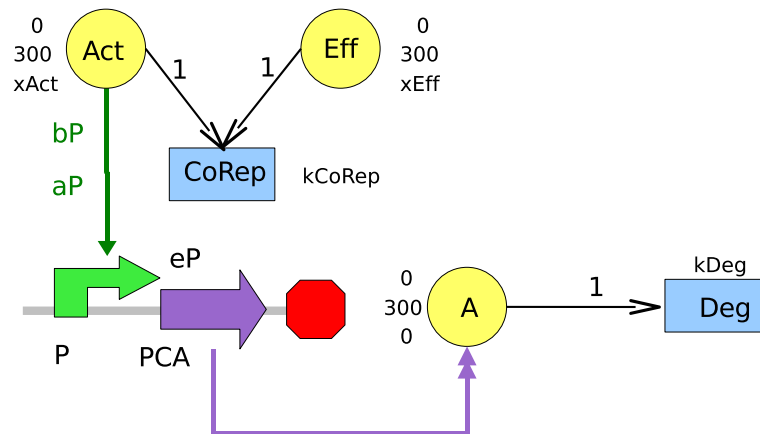


Figure 5.13: Graphical representation of the Gene Activation and Repressible System pattern.

of the repressor *Rep*. The incidence of the repressor population on the expression rate of the downstream coding regions is defined by the binding stochastic constant bP and repression stochastic constant rP . Intuitively, a higher binding rate makes the repressor population more influential in its repressive effect and a higher repression rate enhances the duration of the repressor effect on the promoter.

In Figure 5.13 it is depicted a regulatory scenario with a constitutively *off* promoter P activated by protein *Act* (x_{Eff} initial population is set to zero) and in Table 5.6 is presented the relative PEPA code (PRISM code in Appendix A.0.6). The stochastic simulations in Figure 5.15 represent the system behaviour at different concentrations of the activator *Act*. The incidence of the activator population on the expression rate of the downstream coding regions is defined by the binding stochastic constant bP and the activation stochastic constant aP . The binding rate has the same effect as in the repressor case, a higher activator rate causes an increase in the transcriptional rate.

5.2.3 Inducible and Repressible Systems

One form of regulation of the gene transcription is given by the existence of effectors for the repressor and activator proteins. These effectors are molecules that can bind with the transcriptional regulators and change their functions. Two type of effects are possible: the effector can bind the non-active form of the transcriptional regulator and activate it (*positive co-factor*) or the effector can bind the active form and inactivate it (*negative co-factor*). These forms are present in nature for both repressor and activator transcriptional factors thus determining the existence of two type of systems: *inducible*

and *repressible*. In an *inducible system* the gene expression is *off* (because repressed or not activated), unless there is the presence of an effector molecule (called *inducer*) that can activate the gene expression (by activating the activator or inactivating the repressor). In a *repressible system* the gene expression is *on* (because not repressed or activated), unless there is the presence of an effector molecule (called *corepressor*) that deactivate the gene expression (by deactivating the activator or activating the repressor). Therefore, there exist repressible and inducible systems both in *negative* and in *positive control* scenarios.

In Figure 5.12 we present an *inducible system* in a *negative control* scenario: the repressor molecule is usually active and it is inactivated by binding with the *inducer*. The PEPA code is detailed in Table 5.5 (PRISM code in Appendix A.0.5) and some stochastic simulations in Figure 5.14. The stochastic simulations permit to analyse the behaviour of the repressible system under different conditions: with no presence of the repressor, with few presence of the repressor, with consistent presence of the repressor, with consistent presence of the repressor and few of the inducer, with the consistent presence of the repressor and the inducer. Here we do not report the model of an *inducible system* in a *positive control* scenario, but the model is intuitively similar.

In Figure 5.13 we present a *repressible system* in a *positive control* scenario: the activator molecules is usually active and it is inactivated by binding with the *co-repressor*. The PEPA code is detailed in Table 5.6 (PRISM code in Appendix A.0.6) and stochastic simulations are depicted in Figure 5.15. From the time-course plotting of protein concentration it is possible to compare the system behaviour under different conditions: with no presence of the activator, with few presence of activator, with consistent presence of the activator, with consistent presence of activator and few of the co-repressor, with the consistent presence of the activator and the co-repressor. Here we do not report the model of a *repressible system* in a *negative control* scenario, but the model is intuitively similar.

In biology an example of repressible systems is the *Trp* operon and of inducible systems is the *Lac* operon.

5.2.4 Transcription and Translation Processes

Gene expression is a complex process composed of several sub-steps in between the initiation of transcription and the synthesis of the final functional proteins. Commonly the mechanism is regarded as the sequence of two macro-steps: the *transcription* phase

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(PExp, 1).A(1)$	
$A(i) =$	$(PExp, 1).A(i + 1) + (Deg, i).A(i - 1)$	$0 < i < 300$
$A(300) =$	$(Deg, 300).A(299)$	
$Eff(0) =$	$(null, 1).Eff(0)$	
$Eff(i) =$	$(Ind, i).Eff(i - 1)$	$0 < i \leq 300$
$Rep(0) =$	$(null, 1).Rep(0)$	
$Rep(i) =$	$(P_bind_RepRep, i).Rep(i) +$ $(Ind, i).Rep(i - 1)$	$0 < i \leq 300$
$POp =$	$(PExp, eP).POp +$ $(P_bind_RepRep, bP).POp +$ $(P_unbind_RepRep, rP).POp$	
$Ind =$	$(Ind, kInd).Ind$	
$Deg =$	$(Deg, kDeg).Deg$	

PEPA system definition
$(((((POp \bowtie_{\{PExp\}} A(0)) \bowtie_{\{P_bind_RepRep\}} Rep(xRep)) \bowtie_{\{Ind\}} Eff(xEff)) \bowtie_{\{Ind\}} Ind) \bowtie_{\{Deg\}} Deg$

Number of states: 121002 ($xEff=300, xRep=200$)
Number of transitions: 422001 ($xEff=300, xRep=200$)

Table 5.5: PEPA model definition and information about the associated CTMC for the Gene Repression and Inducible System pattern.

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(PExp, 1).A(1)$	
$A(i) =$	$(PExp, 1).A(i + 1) + (Deg, i).A(i - 1)$	$0 < i < 300$
$A(300) =$	$(Deg, 300).A(299)$	
$Eff(0) =$	$(null, 1).Eff(0)$	
$Eff(i) =$	$(CoRep, i).Eff(i - 1)$	$0 < i \leq 300$
$Act(0) =$	$(null, 1).Act(0)$	
$Act(i) =$	$(P_bind_ActAct, i).Act(i) +$ $(CoRep, i).Act(i - 1)$	$0 < i \leq 300$
$POp =$	$(PExp, eP).POp +$ $(P_bind_ActAct, bP).$ $(PExp, rP).POp$	
$CoRep =$	$(CoRep, kCoRep).CoRep$	
$Deg =$	$(Deg, kDeg).Deg$	

PEPA system definition
$(((((POp_{\{PExp\}} \bowtie A(0))_{\{P_bind_ActAct\}} \bowtie Act(xAct))_{\{CoRep\}} \bowtie Eff(xEff))_{\{CoRep\}} \bowtie CoRep)_{\{Deg\}} \bowtie Deg$

Number of states: 121002 ($xEff=250, xRep=200$)
Number of transitions: 421800 ($xEff=250, xRep=200$)

Table 5.6: PEPA model definition and information about the associated CTMC for the Gene Activation and Repressible System pattern.

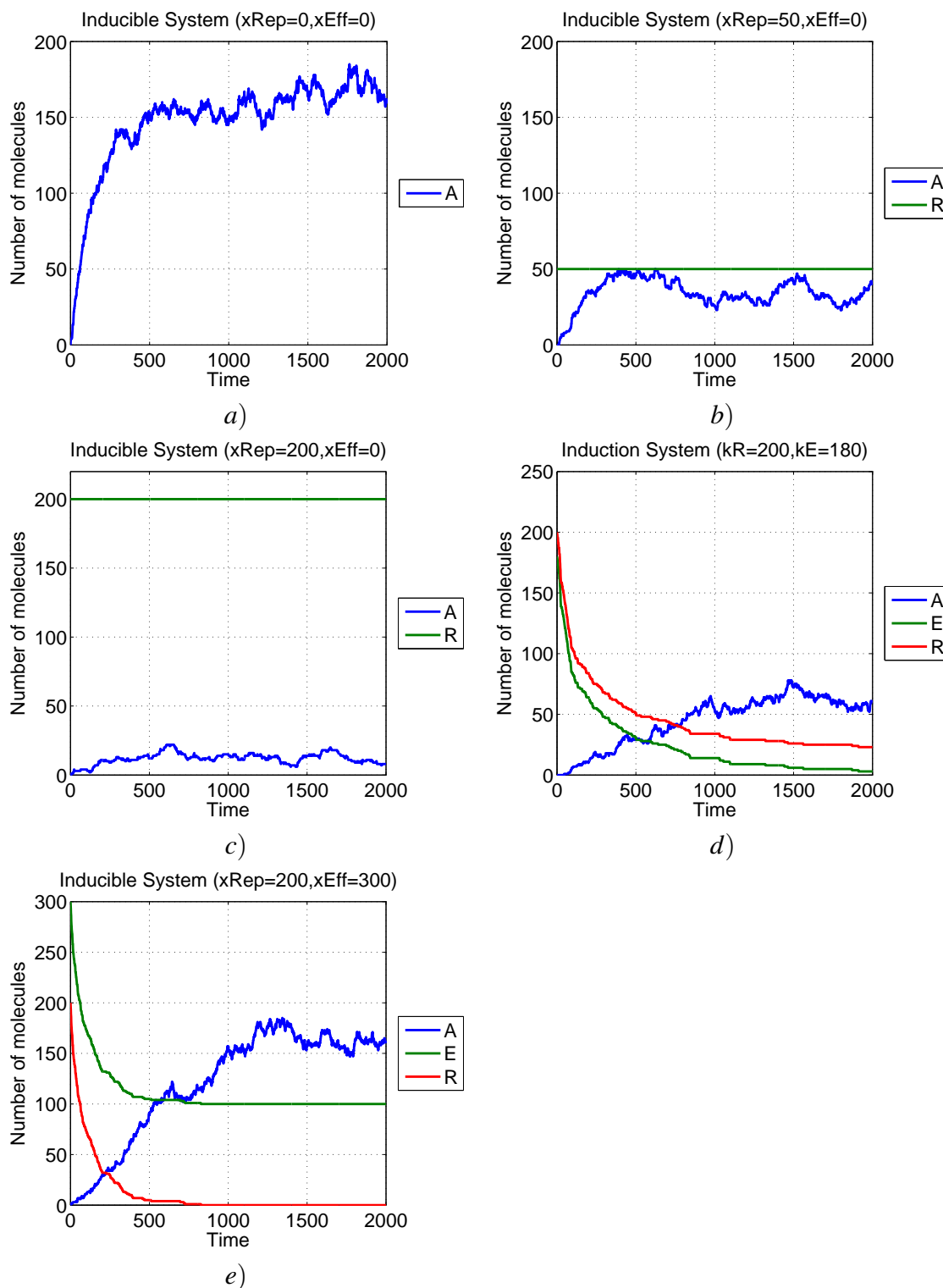


Figure 5.14: Stochastic simulations (single run) for the Gene Repression and Inducible System pattern with $k\text{Deg}=0.005$, $k\text{Ind}=0.00005$, $eP=0.8$, $bP=0.05$ and $rP=0.8$. a) Repressor and effector population is zero. b) Repressor population is 50 molecules. c) Repressor population is 200 molecules. d) Repressor population is 200 molecules and effector population is 180 molecules. e) Repressor population is 200 molecules and effector population is 300 molecules.

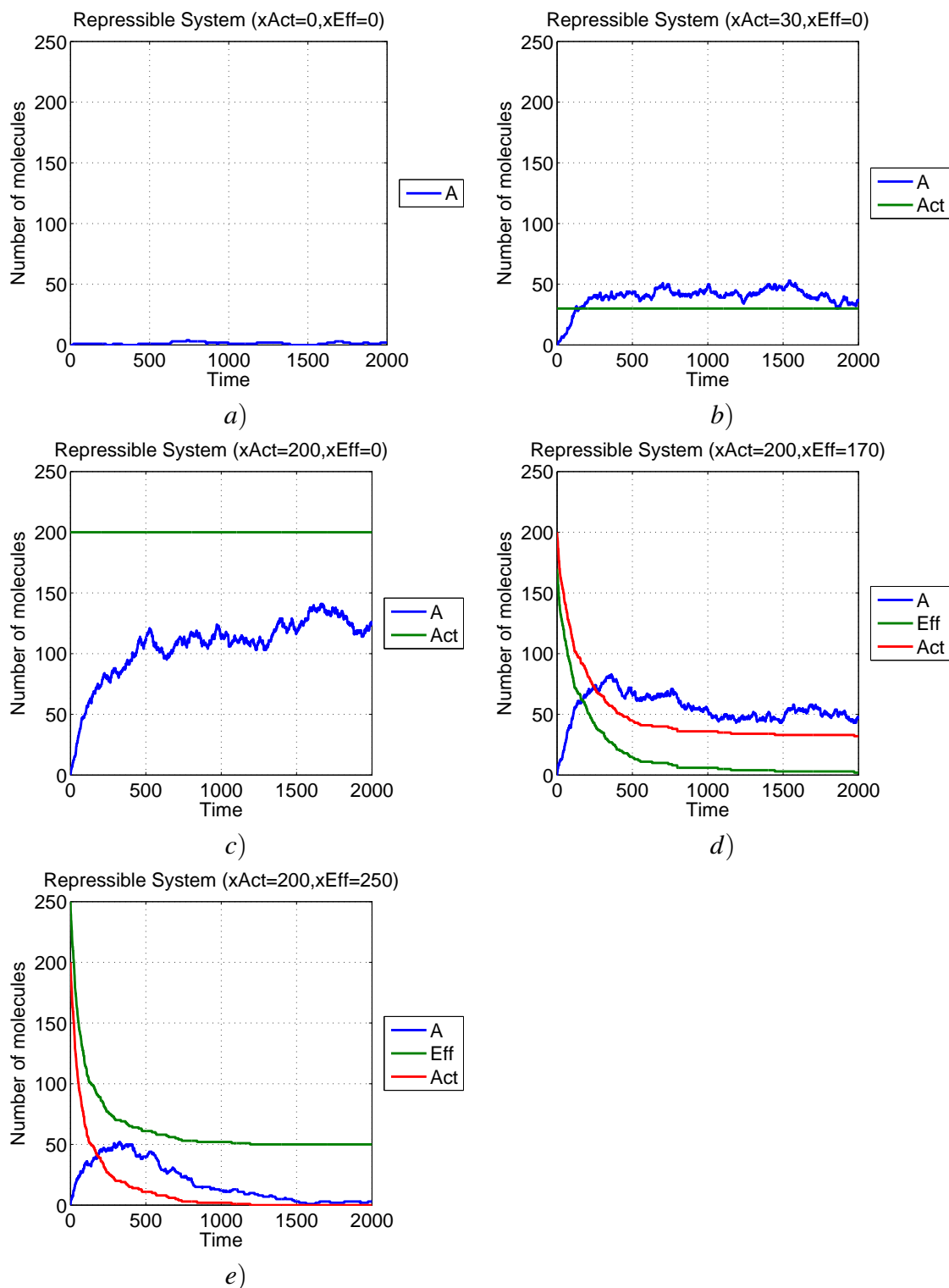


Figure 5.15: Stochastic simulations (single run) for the Gene Activation and Repressible System pattern with $k_{Deg}=0.005$, $k_{CoRep}=0.00005$, $eP=0.009$, $bP=0.01$ and $aP=0.8$. a) Activator and effector population is zero. b) Activator population is 30 molecules. c) Activator population is 200 molecules. d) Activator population is 200 molecules and effector population is 170 molecules. e) Activator population is 200 molecules and effector population is 250 molecules.

and the *translation* phase. *Transcription* is the process by which the genetic information contained in stable DNA molecule is copied by specific enzymes into a complementary, more volatile, nucleotide RNA strand. The RNA information is then converted into an intermediate mRNA form, with biological mechanisms that vary widely between prokaryotes and eukaryotes. *Translation* is the process, occurring in the cytosol, by which the ribosome complexes interact with the mRNA molecules in order to assemble the proteins. Ribosomes use mRNA genetic information as a template for recruiting the right sequence of amino acids entities, according to specific *Genetic Code* rules.

Important forms of gene expression control involve mechanisms at the level of mRNA populations, therefore it might be necessary to consider them in systems' modelling. Among others, aspects that regulate mRNA populations are the degradation rate, the affinity of binding with the ribosome, the velocity of translation and the interaction with other RNA molecules (for example *RNA interference* mechanisms).

In Figure 5.16 it is depicted the situation of two protein coding regions controlled by the same promoter region in which the transcription and translation processes have been modelled specifically. The direct products of the protein coding regions are respective mRNA populations, subjected to an individual degradation and translation process. Therefore the mRNA populations are replenished at the same rate by the promoter control, but differences in the final proteins concentrations can be obtained setting different degradation and translation stochastic constants. In Table 5.7 is presented the automatically generated PEPA code (PRISM code in Appendix A.0.7). In Figure 5.17 are shown differences in mRNA and proteins populations behaviour under different settings of the degradation and translation constants. To note that, in this modelling approach, *RNA interference* mechanisms might be modelled as reagent-product reactions among mRNA populations.

5.3 Common patterns in membrane and transport networks

The cell structure is composed of several subarea units that are separated by membranes. The plasma membrane surrounds the cell and separates its internal environment from the external world, other membranes divide the internal area defining enclosed spaces or compartments. Compartments primary function is to maintain a

PEPA processes definitions		
Name	Body	Conditions
$A(0) =$	$(TransA, 1).A(1)$	
$A(i) =$	$(TransA, 1).A(i+1) + (DegA, i).A(i-1)$	$0 < i < 100$
$A(100) =$	$(DegA, 100).A(99)$	
$B(0) =$	$(TransB, 1).B(1)$	
$B(i) =$	$(TransB, 1).B(i+1) + (DegB, i).B(i-1)$	$0 < i < 100$
$B(100) =$	$(DegB, 100).B(99)$	
$mA(0) =$	$(PExp, 1).mA(1)$	
$mA(i) =$	$(PExp, 1).mA(i+1) + (DegmA, i).mA(i-1) + (TransA, i).mA(i-1)$	$0 < i < 100$
$mA(100) =$	$(DegmA, 100).mA(99) + (TransA, 100).mA(99)$	
$mB(0) =$	$(PExp, 1).mB(1)$	
$mB(i) =$	$(PExp, 1).mB(i+1) + (DegmB, i).mB(i-1) + (TransB, i).mB(i-1)$	$0 < i < 100$
$mB(100) =$	$(DegmB, 100).mB(99) + (TransB, 100).mB(99)$	
$POp =$	$(PExp, eP).POp$	
$DegA =$	$(DegA, kDegA).DegA$	
$DegB =$	$(DegB, kDegB).DegB$	
$DegmA =$	$(DegmA, kDegmA).DegmA$	
$DegmB =$	$(DegmB, kDegmB).DegmB$	
$TransA =$	$(TransA, kTransA).TransA$	
$TransB =$	$(TransB, kTransB).TransB$	

PEPA system definition
$((((((((POp \bowtie_{\{PExp\}} mA(0)) \bowtie_{\{PExp\}} mB(0)) \bowtie_{\{TransA\}} A(0)) \bowtie_{\{TransB\}} B(0)) \bowtie_{\{TransA\}} TransA) \bowtie_{\{TransB\}} TransB) \bowtie_{\{DegmA\}} DegmA) \bowtie_{\{DegmB\}} DegmB) \bowtie_{\{DegA\}} DegA) \bowtie_{\{DegB\}} DegB$

Number of states: 104060401

Number of transitions: 718150400

Table 5.7: PEPA model definition and information about the associated CTMC for the Transcription and Translation Processes pattern.

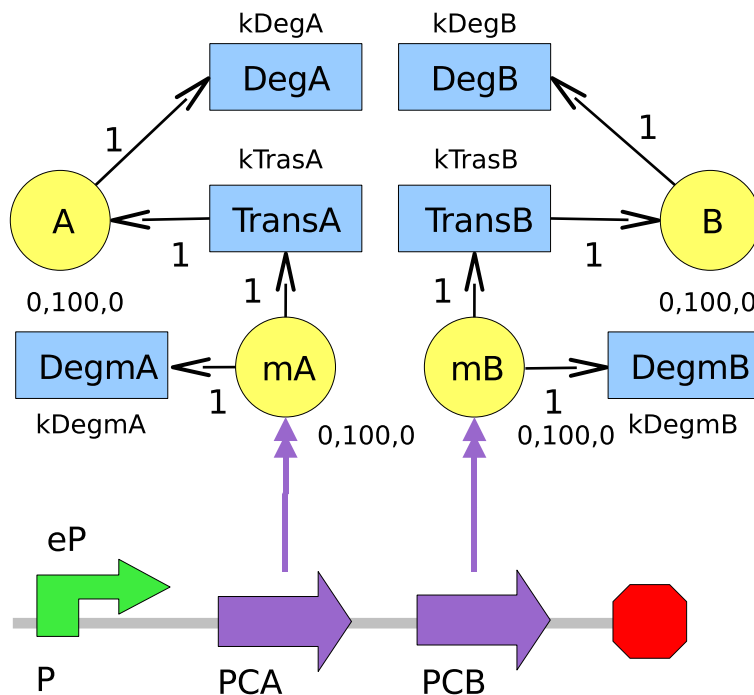


Figure 5.16: Graphical representation of the Transcription and Translation Processes pattern.

biochemical environment different from the external in order to provide particular favorable conditions for specific reactions or mechanisms. Membranes function is not only to divide spaces but also to act as selective filter for molecules passage. A membrane is usually a semipermeable lipid bilayer, made of a double layer of lipid-class molecules with the possible presence of particular structures that act as channels. Different mechanisms for the passage of molecules across membrane structures are known, such as *osmosis* or *vesicular transport*. Moreover, processes that occur on the membrane might be of fundamental importance, as in the case of interactions between external molecules and membrane receptors.

In certain biological scenarios, it might be necessary to include compartments information in the cell models. In the context of our modelling framework, the presence of compartments can be modelled intuitively by defining different species for the same kind of protein placed in different ambient. Since the compartments act as close areas for biochemical reactions, this solution permits to specify reactions for individual molecules that are in a compartment instead of another one. This approach permits the representation of *static compartment structures*. In Section 5.3.1 is presented the modelling pattern for *passive transport* among compartments, in Section 5.3.2 is re-

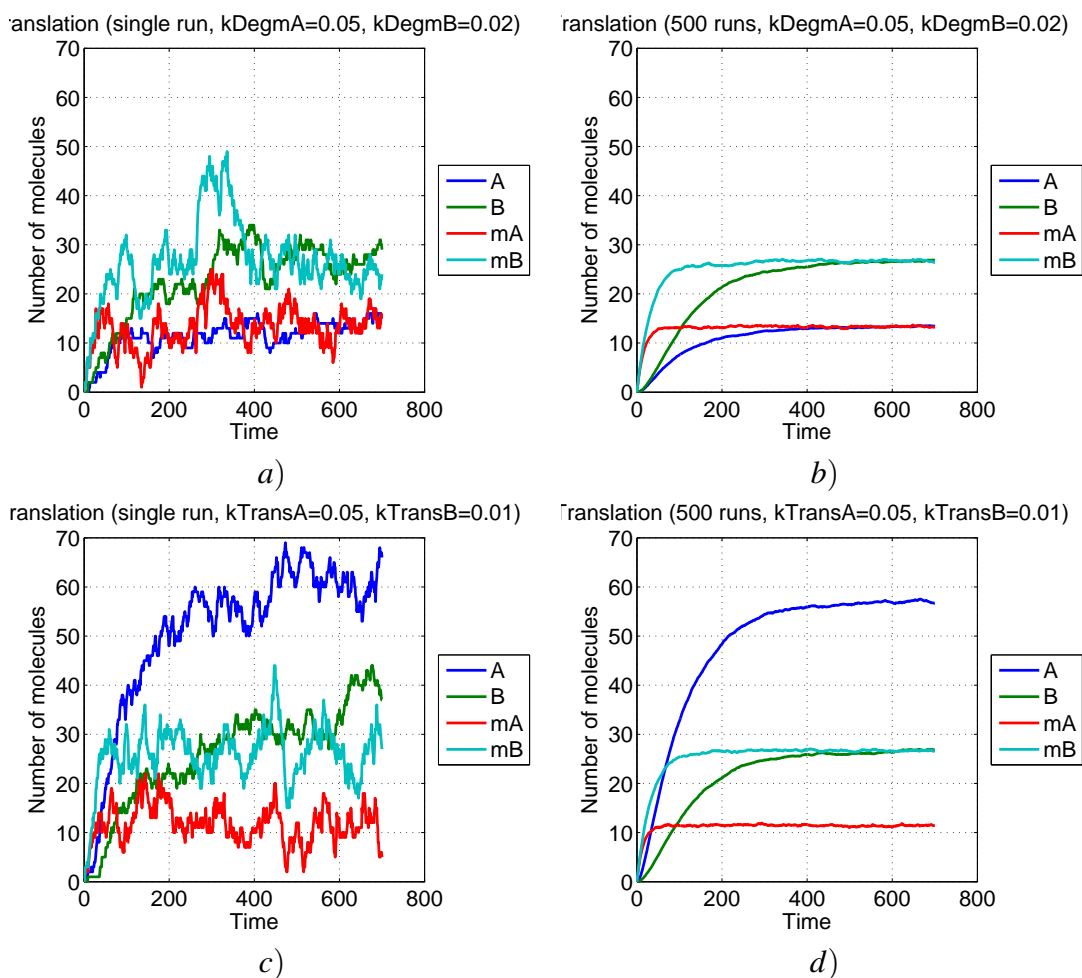


Figure 5.17: Stochastic simulations for the Transcription and Translation pattern with (if not explicitly stated on the top of each graph) $k_{DegA}=k_{DegB}=0.01$, $k_{DegmA}=k_{DegmB}=0.02$, $k_{TrasA}=k_{TrasB}=0.01$. a) Single simulation run with the two proteins having different mRNA degradation rates. b) Average of 500 simulation run with the two proteins having different mRNA degradation rates. c) Single simulation run with the two proteins having different translation rates. d) Average of 500 simulation run with the two proteins having different translation rates.

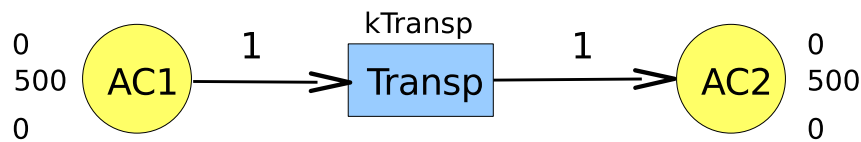


Figure 5.18: Graphical representation of the Passive Transport pattern.

ported the approach for the transport using *vescicule*. In Section 5.3.3 are described guidelines for representing interactions between receptors and ligands.

5.3.1 Passive Transport

The lipid bilayer that composes membranes is usually permeable to external and internal small molecules that can cross compartments by osmosis. This transport system is termed *passive* because the cell does not spend energy in the process. This is the most simple mechanism for which external signals and nutrients can enter into and exit from the cell. Sometimes the process is selective and permits the passage only to molecules that bind with special polarised proteins. This proteins are able to separate the bilayer structure and carry on the targeting molecule.

In Figure 5.18 we present a simple modelling scenario in which protein A can be present in two different compartments, C1 and C2. The protein populations is represented as two different species and a reaction is responsible of simulating the osmosis transfer of individuals from one compartment to the other. The probability of crossing is higher when the population is higher. In Table 5.8 there is the automatic generated PEPA code and in Appendix A.0.8 the correspondent PRISM code. In Figure 5.19 are proposed stochastic simulations of population behaviour when the transport rate is changed.

5.3.2 Vesicular Transport

An important mechanism for transport inside and outside the cell is the ability of membranes to expell and engulf vesicule carrying proteins. The *exocytosis* process start with the formation of a cavity in the membrane that then pinches off a spherical vesicle containing the material to be transported. The complementary process is called *phagocytosis* or *endocytosis* and proceeds by the attachment of the vesicule with the membrane border and concludes with the taking up of the contained material by the internal environment. The *vesicular transport* is particular important for large mole-

PEPA processes definitions		
Name	Body	Conditions
$AC1(0) =$	$(null, 1).AC1(0)$	
$AC1(i) =$	$(Transp, i).AC1(i-1)$	$1 \leq i \leq 500$
$AC2(i) =$	$(Transp, 1).AC2(i+1)$	$0 \leq i \leq 499$
$AC2(500) =$	$(null, 1).AC2(500)$	
$Transp =$	$(Transp, kTransp).Transp$	

PEPA system definition
$(AC1(500) \underset{\{Transp\}}{\bowtie} AC2(0)) \underset{\{Transp\}}{\bowtie} Transp$

Number of states: 201
Number of transitions: 201

Table 5.8: PEPA model definition and information about the associated CTMC for the Passive Transport pattern.

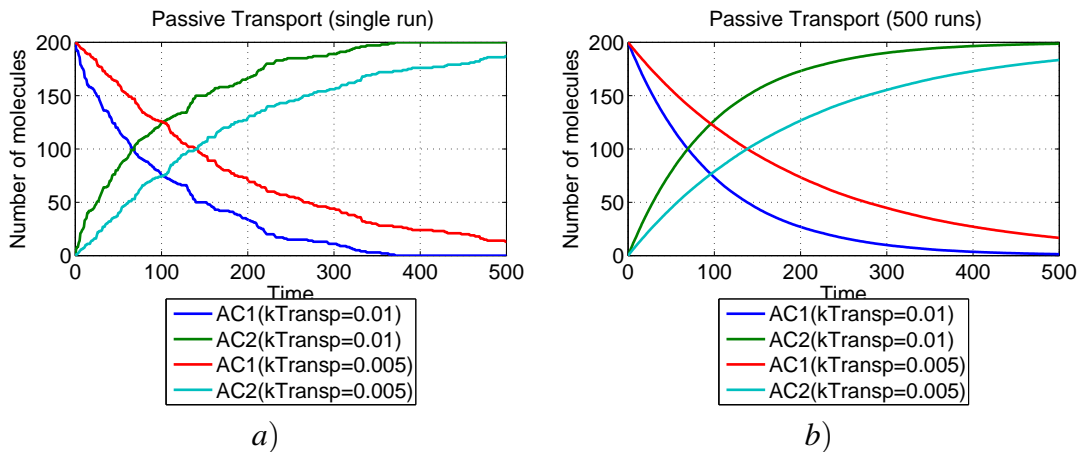


Figure 5.19: Stochastic simulations for the Passive Transport pattern. a) Single simulation run with different values for kTransp. b) Average of 500 simulation runs with different values for kTransp.

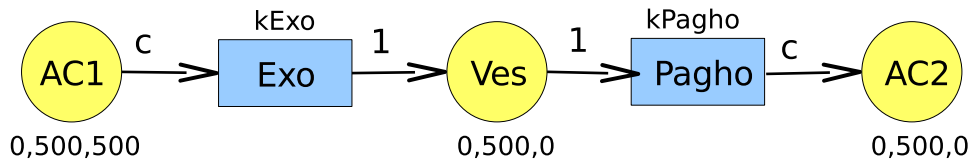
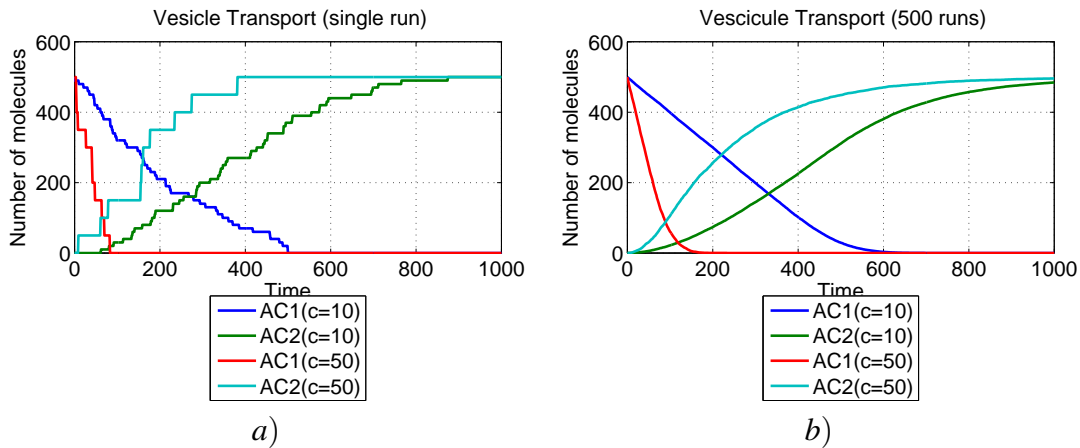


Figure 5.20: Graphical representation of the Vesicular Transport pattern.

Figure 5.21: Stochastic simulations for the Vesicular Transport pattern with $kExo=0.1$, $kPagho=0.005$. a) Single simulation run with different values for parameter c . b) Average of 500 simulation runs with different values for parameter c .

cues (i.e. proteins) that cannot cross the membrane by *osmosis*. The process is termed *active* because the cell spends energy in performing the transport.

In Figure 5.20 we present a possible approach in modelling this kind of transport mechanism. The same protein present in two different compartments $C1$ and $C2$ is represented by two different species. The movement through a vesicle is modelled as a third species Ves , in which each individual of the population represent a vesicle traveling from $C1$ to $C2$ and carrying c A proteins. The automatically generate PEPA model (in Tabel 5.9) has been slightly modified because *mass action* law is not a reasonable assumption for the rate of vesicle creation. The creation rate is controlled only by the stochastic constant $kExo$.

Using model checking formula $P_{\infty?}[trueU^{[t,t]}(ves = i)]$ is possible to investigate “what is the probability of there being i vesicle travelling from A to B at time T ”. Figure 5.22 shows the resulting distribution of Ves population with i ranging from 0 to 15 and time ranging from 0 to 500 with a discretized step of 50 units.

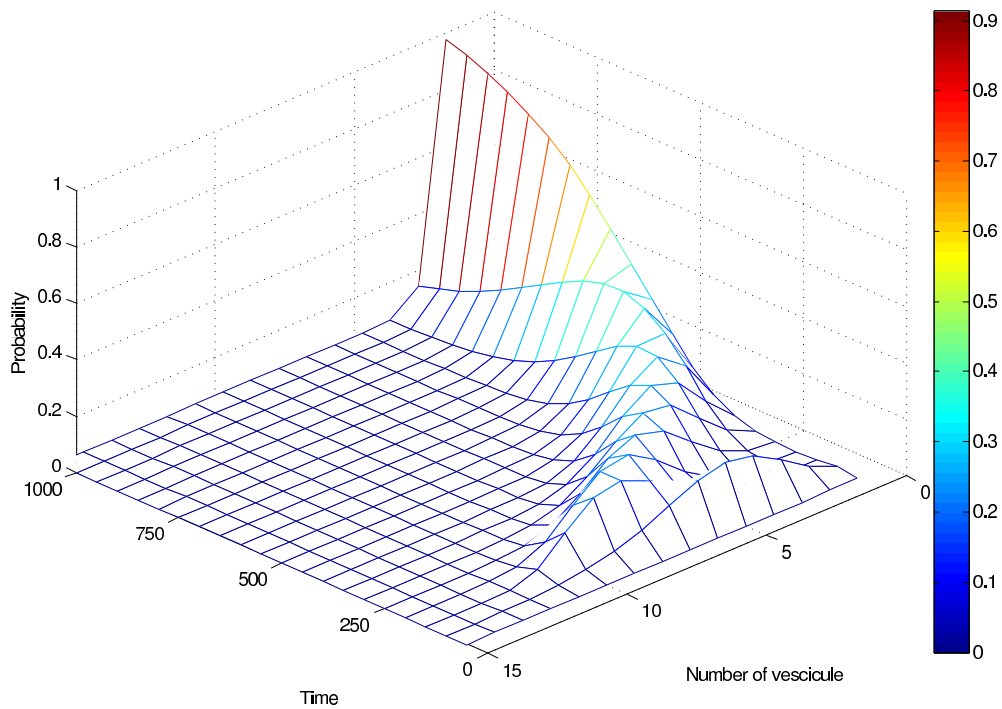


Figure 5.22: Plot representing the probabilistic distribution of the number of vesicle travelling from compartment A to compartment B in time in the Vesicular Transport pattern ($k_{Exo}=0.1$, $k_{Pagho}=0.005$, $c=10$). The probability of having i vesicle travelling at time t has been calculated using model checking techniques. Time has been discretized with a step of 50 units from 0 to 500.

PEPA processes definitions		
Name	Body	Conditions
$AC1(0) =$	$(null, 1).AC1(0)$	
$AC1(i) =$	$(Exo, 1).AC1(i - c)$	$c \leq i \leq 500$
$AC2(i) =$	$(Pagho, 1).AC2(i + c)$	$0 \leq i \leq 500 - c$
$AC2(500 - i) =$	$(null, 1).AC2(500 - i)$	$0 < i \leq c$
$Ves(0) =$	$(Exo, 1).Ves(1)$	
$Ves(i) =$	$(Exo, 1).Ves(i + 1) + (Pagho, i).Ves(i - 1)$	$0 \leq i \leq 499$
$Ves(500) =$	$(Pagho, 500).Ves(499)$	
$Exo =$	$(Exo, kExo).Exo$	
$Phago =$	$(Phago, kPhago).Phago$	

PEPA system definition
$(AC1(500) \underset{\{Exo\}}{\boxtimes})Ves \underset{\{Pagho\}}{\boxtimes} AC2(0)$

Number of states: 1326
Number of transitions: 2551

Table 5.9: PEPA model definition and information about the associated CTMC for the Vesicular Transport pattern.

5.3.3 Receptor-Ligand Interactions

Cells have developed several mechanisms for handling external signals and processing the adequate response. Two are the broad categories in which the signals are carried inside the cell: molecules carrying the informations may penetrate the cell by *passive transport* and bind to internal receptors or they can be perceived by external *transmembrane receptors*. The latter case is of principal importance in the modelling of *signalling pathways networks*. The molecule carrying information is called *ligand* and it can bind specifically with a particular receptor domain. When the interaction occurs, the transmembrane receptor is affected by a conformation change in its internal part. This change triggers an activation signal inside the cell through a reaction cascade. Finally the receptor comes back to the susceptible form. In order to increase or decrease the sensitivity to signals during stimulations, cells are able to regulate the population of their receptors and their state of activity.

In Figure 5.23 we propose a model of these dynamics. The susceptible receptor population is represented by R_s and the active receptor population by R_a . A susceptible receptor can switch to an active state by binding to a ligand individual L . When active, a receptor can carry on the signal by activating several protein $PAct$. With a certain

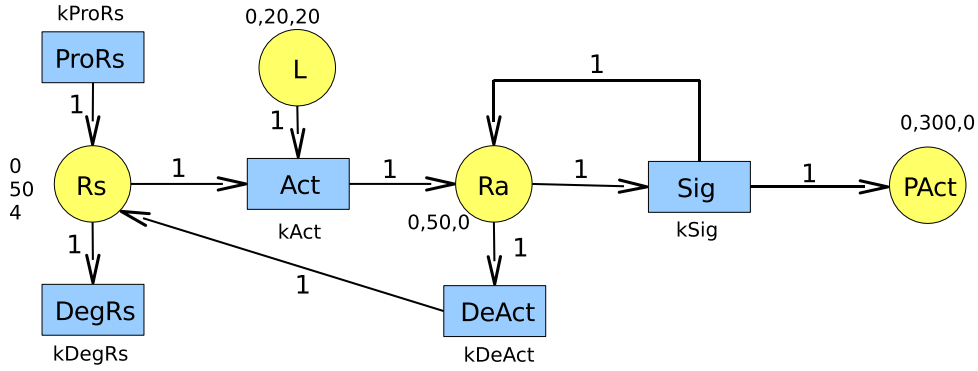


Figure 5.23: Graphical representation of the Receptors-Ligand Interactions pattern.

PEPA processes definitions		
Name	Body	Conditions
$Rs(0) =$	$(ProRs, 1).Rs(1) + (DeAct, 1).Rs(1)$	
$Rs(i) =$	$(ProRs, 1).Rs(i+1) + (DegRs, i).Rs(i-1) + (Act, i).Rs(i-1) + (DeAct, 1).Rs(i+1)$	$1 \leq i \leq 49$
$Rs(50) =$	$(DegRs, 50).Rs(49) + (Act, i).Rs(49)$	
$Ra(0) =$	$(Act, 1).Ra(1)$	
$Ra(i) =$	$(Act, 1).Ra(i+1) + (Sig, i).Ra(i) + (DeAct, i).Ra(i-1)$	$1 \leq i \leq 49$
$Ra(50) =$	$(Sig, 50).Ra(50) + (DeAct, 50).Ra(49)$	
$L(0) =$	$(null, 1).L(0)$	
$L(i) =$	$(Act, i).L(i-1)$	$1 \leq i \leq 20$
$PAct(i) =$	$(Sig, 1).PAct(i+1)$	$0 \leq i \leq 299$
$PAct(300) =$	$(null, 1).PAct(300)$	
$ProRs =$	$(ProRs, kProRs).ProRs$	
$DegRs =$	$(DegRs, kDegRs).DegRs$	
$Act =$	$(Act, kAct).Act$	
$DeAct =$	$(DeAct, kDeAct).DeAct$	
$Sig =$	$(Sig, kSig).Sig$	

PEPA system definition
$((((((((Rs(4) \bowtie_{\{Act\}})Ra(0)) \bowtie_{\{Act\}} L(20)) \bowtie_{\{Sig\}} PAct(0)) \bowtie_{\{ProRs\}} ProRs)$ $\bowtie_{\{DegRs\}} DegRs) \bowtie_{\{Act\}} Act) \bowtie_{\{DeAct\}} DeAct) \bowtie_{\{Sig\}} Sig$

Number of states: 8048565
Number of transitions: 37907700

Table 5.10: PEPA model definition and information about the associated CTMC for the Receptor-Ligand Interactions pattern.

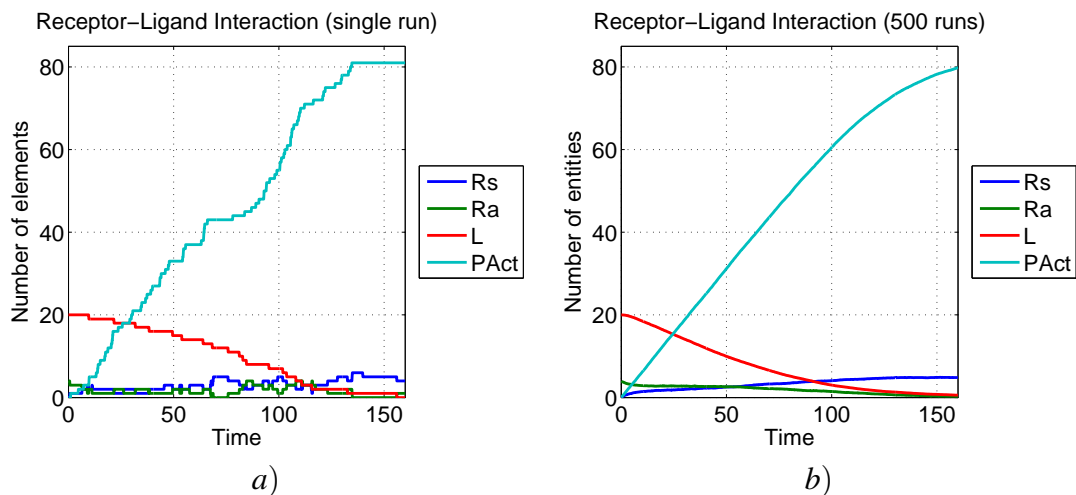


Figure 5.24: Stochastic simulations for the Receptor-Ligand Interactions pattern with $kProRs=0.04$, $kDegRs=0.01$, $kAct=0.008$, $kSig=0.8$, $kDeAct=0.08$. a) Single simulation run. b) Average of 500 simulation runs.

probability each receptor can switch back to the inactive form. The susceptible population is in constant fluctuation, the number regulated by a creation and degradation rate. In Table 5.10 it is presented the automatic generate PEPA model (PRISM code in Appendix A.0.10). In Figure 5.24 are shown stochastic simulations of the receptor response when ligands are available for binding.

5.4 Case study

In this section, our modelling approach is applied to a real case study, the Synthetic Biology device designed and constructed by the Edinburgh Team for the iGEM 2006 competition (see Section 2.1). Along with the graphical notation representation and the process algebra model, we present the biological description and characterisation of the standard biological parts (from the *Registry of Biological Standard Parts*) that have been used in the construction.

5.4.1 Edinburgh iGEM 2006: the Arsenic Biosensor

As part of the *International Genetically Engineered Machine* competition (iGEM) 2006, several students from the University of Edinburgh developed a Synthetic Biology system able to sense and report the presence of arsenic in the water [29]. The problem of poisoning caused by the presence of arsenic in the drinking water is a major issue

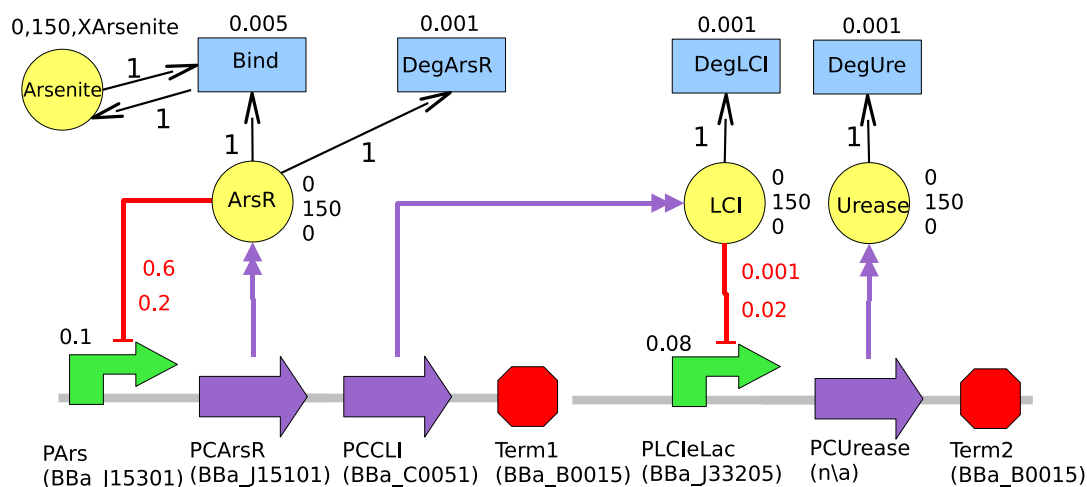


Figure 5.25: Graphical representation of the Arsenic Biosensor system. The concentration of Urease depends on the arsenate initial concentration $xArsenite$.

Part (code)	Description
<i>PArs</i> (BBa_J15301)	This part is the promoter region of the Escherichia coli chromosomal <i>ars</i> promoter. It is repressed by <i>ArsR</i> in the absence of arsenate or arsenite.
<i>PCArSR</i> (BBa_J15101)	This part is the Escherichia coli chromosomal <i>arsR</i> coding sequence, which encodes the <i>ArsR</i> repressor. The repressor molecule binds to the <i>ars</i> promoter and represses it in the absence of arsenate or arsenite.
<i>PCCLI</i> (BBa_C0051)	This is the coding region part for the <i>cI</i> repressor based on <i>cI</i> repressor from bacteriophage lambda. <i>cI</i> repressor binds to the <i>cI</i> regulator.
<i>PLCeLac</i> (BBa_J33205)	This part is a hybrid promoter designed to be repressed by either lambda <i>cI</i> or <i>LacI</i> .
<i>PCUrease</i> (n/a)	This part has not been yet characterised in the form of bio-brick. It is the urease gene cluster containing <i>ureABC</i> gene regions. It codes for proteins that are able to increase the PH of cells' solution.
<i>Term1,2</i> (BBa_B0015)	This part is a transcription terminator, used to stop the transcription of a DNA region.

Table 5.11: The table presents the Registry description of each one of the seven standard DNA parts used in constructing the Arsenic Biosensor system. The first attribute is the name we used in the graphical description, in between brackets is reported their identifier code in the *Registry of Standard Biological Parts*.

in several third world countries, therefore the system construction was undertaken in order to produce a cheap and reliable detection system. The output of the system is a pH change, as response of a dangerous concentration of arsenic in the water.

This whole-cell arsenic biosensor relies upon the chromosomal arsenic detection system of *Escherichia coli*, consisting of the *ars* promoter and the ArsR repressor protein, which response to arsenic in the form of arsenate or arsenite anions. To generate a pH change, the system relies on the expression of urease by the ureABC gene cluster. In its internal circuit design, the biosensor uses the lambda repressor system of λ phage to activate or inactivate the production of urease. The system was constructed as composition of several standard biological parts, parts that were taken from the *Registry of Biological Standard Parts* or biobricked by the team. A description of the biological characteristics and functions of each part is proposed in Table 5.11.

Figure 5.25 presents the internal circuit design of the system. There are two operon structures, called *Device 1* and *Device 2*. *Device 1* is composed of the *ars* promoter region followed by a Ars repressor coding region and a cI repressor coding region. When there is no presence of arsenate, the promoter region is continuously repressed by the ArsR repressor, therefore the concentration of cI repressor is minimal. When arsenate is present in the water, the ArsR repressor binds with higher affinity to arsenite molecules, therefore allowing the expression of cI repressor. *Device 2* responds to changes in cI concentration. Under the assumption that there is no lac in the system, the promoter is *on* (active) and therefore urease is present in great quantity in the system. When cI concentration builds up, the hybrid Lac and cI promoter is repressed, therefore the expression of Urease is blocked and finally the pH decreases.

The system reported here was constructed as a proof of concept example since the exact calibration in the lab to tune every biological process properly is supposed to take at minimum several months of work. We therefore propose here a model for which rates and population quantities have been taken as reasonable assumptions or inferred from the once in the ODE model realised by the Edinburgh team. The aim of this modelling example is mainly to prove the expressiveness of our modelling approach, more than to produce an exact quantitative and temporal prediction of the system behaviour. Table 5.12 presents the PEPA model (PRISM code in Appendix A.0.11). Figure 5.26 presents the time-course behaviour of populations under three different initial concentrations: with no, few and high level of arsenic present in the water. As the graphs show, the model predicts correctly the behaviour of repressor proteins and their effects on the promoter regions, as well as in predicting the change in Urease population.

PEPA processes definitions		
Name	Body	Conditions
$PLCIeLacOp =$	$(PLCIeLacExp, 0.08).PLCIeLacOp +$ $(PLCIeLac_bind_LCIRep, 0.001).$ $(PLCIeLac_unbind_LCIRep, 0.02).PLCIeLacOp$	
$PArsOp =$	$(PArsExp, 0.1).PArsOp +$ $(PArs_bind_ArsRRep, 0.6).$ $(PArs_unbind_ArsRRep, 0.2).PArsOp$	
$ArsR(0) =$ $ArsR(i) =$ $ArsR(150) =$	$(PArsRExp, 1).ArsR(1)$ $(Bind, i).ArsR(i - 1) +$ $(DegArsR, i).ArsR(i - 1) +$ $(PArsRExp, 1).ArsR(i + 1) +$ $(PArs_bind_ArsRRep, i).ArsR(i)$ $(Bind, 150).ArsR(149) +$ $(DegArsR, 150).ArsR(149) +$ $(PArs_bind_ArsRRep, 150).ArsR(150)$	$1 \leq i \leq 149$
$LCI(0) =$ $LCI(i) =$ $LCI(150) =$	$(PCCLIExp, 1).LCI(1)$ $(PCCLIExp, 1).LCI(i + 1) +$ $(DegLCI, i).LCI(i - 1) +$ $(PCCLI_bind_LCIRep, i).LCI(i)$ $(DegLCI, 150).LCI(149) +$ $(PCCLI_bind_LCIRep, 150).LCI(150)$	$1 \leq i \leq 149$
$Urease(0) =$ $Urease(i) =$ $Urease(150) =$	$(PLCIeLacExp, 1).Urease(1)$ $(PLCIeLacExp, 1).Urease(i + 1) +$ $(DegUre, i).Urease(i - 1)$ $(DegUre, 150).Urease(149)$	$1 \leq i \leq 149$
$Arsenite(xArsenite) =$	$(Bind, xArsenite).Arsenite(xArsenite)$	
$DegArsR =$	$(DegArsR, 0.001).DegArsR$	
$DegLCI =$	$(DegLCI, 0.001).DegLCI$	
$DegUre =$	$(DegUre, 0.001).DegUre$	
$Bind =$	$(Bind, 0.005).Bind$	

PEPA system definition
$((((((((Arsenite(xArsenite) \bowtie_{\{Bind\}} ArsR(0)) \bowtie_{\{PArs_bind_ArsRRep, PArsExp\}} PArsOp) \bowtie_{\{PArsExp\}} LCI(0))$ $\bowtie_{\{PLCIeLac_bind_LCIRep\}} PLCIeLac) \bowtie_{\{PLCIeLacExp\}} Urease(0))$ $\bowtie_{\{DegArsR\}} DegArsR) \bowtie_{\{DegLCI\}} DegLCI) \bowtie_{\{DegUre\}} DegUre) \bowtie_{\{Bind\}} Bind)$

Number of states: 13771804

Number of transitions: 82129504

Table 5.12: PEPA model definition and information about the associated CTMC for the Arsenic Biosensor system.

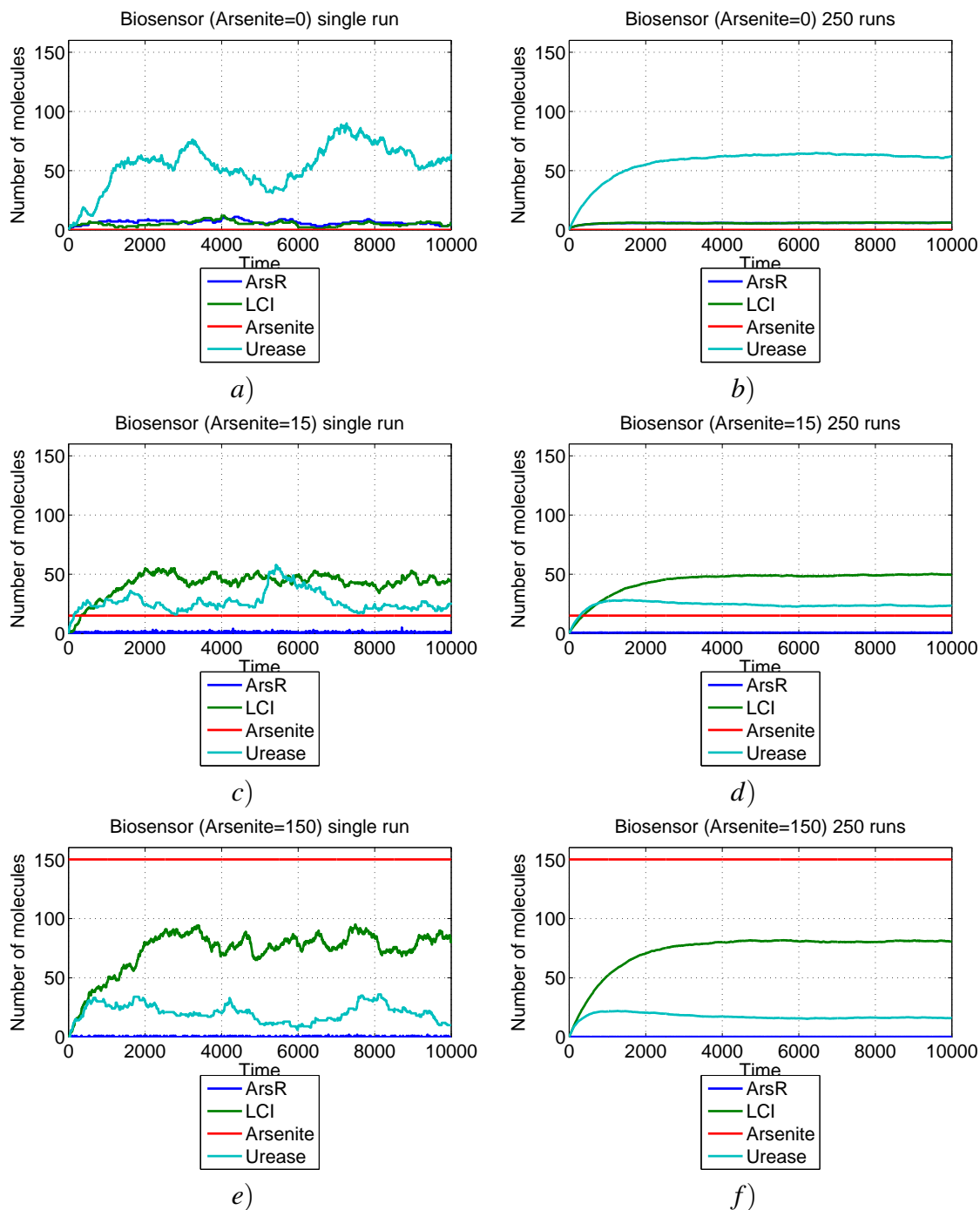


Figure 5.26: Stochastic simulations for the Arsenic Biosensor systems model. a) Arsenite concentration is zero (single run). b) Arsenite concentration is zero (average of 250 runs). c) Arsenite concentration is low (single run). d) Arsenite concentration is low (average of 250 runs). e) Arsenite concentration is high (single run). f) Arsenite concentration is high (average of 250 runs)

Chapter 6

Conclusions

In this final chapter we summarise the outcomes of this thesis work. Section 6.1 presents an overview of the performed work and a critical discussion on the results obtained. Section 6.2 proposes directions for future works. Section 6.3 concludes with some final considerations.

6.1 Concluding remarks and critical discussion

This project focused on proposing and evaluating the stochastic process algebra formalism as suitable framework for the design and analysis of biological systems in the context of Synthetic Biology [1].

The first part of the work consisted in the developing of a graphical notation able to represent unicellular biological systems whose genome has been synthetically modified using standard and composable DNA parts. The notation is composed of nine symbols that can be linked together in order to form pattern compositions. The symbols represent biological entities (*biochemical species, promoters, protein coding regions, transcription terminators*) and biological interactions among them (*promoter repression, promoter activation, participation in a reaction, coding for a specific species*). Symbols are enriched with quantitative information about the cardinality of biological entities, i.e. size of the populations, and time rate values for processes, i.e. gene expression. Symbols that represent DNA parts and transcriptional regulatory effects are identical to the ones adopted in the *Library of Standard Biological Parts* [3], therefore intuitive for the discipline's research community. Symbols adopted for reactions specification are similar to the ones in the Petri Nets graphical notation, therefore intu-

itive for computer scientists. The graphical notation elements have been mapped into a typed data structure and thus can be saved and accessed in a formal fashion. We have shown the notation to be informative enough for the representation of important Synthetic Biology scenarios.

The second part regarded the definition of an automatic conversion from the graphical representation of systems to models written using the stochastic process algebra PEPA (Performance Evaluation Process Algebra). The adopted modelling solution follows the 'cells-as-computation' [6] paradigm and a mechanistic approach. This means that we represent individual components and processes as abstraction of real biological mechanisms and then we derive the system behaviour as property emerging from the interactions between elements. This approach brings some characteristic advantages of process algebras to our work, such as the flexibility in selecting the correct level of detail and compositionality of elements, properties that facilitate the design and hypothesis testing of systems [16]. In this paradigm, we modelled biological entities as discrete quantities and their behaviour as probabilistic processes, according to the emerging general consensus that regards stochasticity and noise as important characteristics of biochemical systems mechanisms. In modelling species populations, we adopted an hybrid (and novel) approach in between the individual representation of each single molecules, as adopted by the majority of process algebra modelling solutions [14], and the discrete level concentration solutions applied in past PEPA works [30], [31]. We represented populations as discrete levels of molecules with a unitary step (a single molecule), therefore facilitating a possible future introduction of our work in both approaches.

In developing the models, we relied on features common to the most of process algebra dialects (sequences of exponentially distributed timed actions, choice operator and cooperation operator) in order to be representative for the formalism family. Our model style relies on multi-way synchronization of processes, a feature of PEPA that is not available in some process algebras but that can usually be mimicked as sequences of pairwise interactions.

We showed that each PEPA model defines an underlying Continuous Time Markov Chain stochastic process and discussed the generated state and transition space with regards to biological mechanisms they represent in the system. In particular we showed that, when limited to elementary biochemical systems, the underlying CTMC is sound, as exact derivation of the Master Equation, in the terms firstly proposed by Gillespie

[24]. In handling gene expression and transcriptional factor regulation aspects, the resulting stochastic process shows simplified assumptions on DNA-molecules binding mechanism and time rates. The level of detail appears to be in any case sufficient for representing the macro effects of such regulatory processes and can be refined in future works.

The definition of an additional automatic conversion from PEPA models to probabilistic model checker PRISM models was developed in order to enable various types of system analysis, such as *steady state analysis*, *probabilistic model checking* and *stochastic simulations*.

Finally, the modelling approach has been applied in the representation and analysis of several biological examples. We showed how to represent patterns that compose some of the most important biological networks usually of interest of Synthetic Biology investigation. In the context of *biochemical signalling networks* we presented modelling solutions for protein degradation, reversible and irreversible reactions and enzymatic reactions. Regarding *synthetic gene networks* we presented solutions for the modelling of gene expression, transcriptional regulation effects of activation and repression, repressible and inducible systems and the specification of transcriptional and translational processes. In considering *membrane and transport networks* we constructed models for passive and active (vesicular) transport between compartments and a model of receptors-ligand interactions. For each example we reported its graphical representation, the automatic generated PEPA model and the related translation in PRISM language. For each example we ran stochastic simulations of the systems using the Gillespie's Algorithm and showed the behaviour of species populations in time under different model settings (i.e. varying initial populations or reaction rates). When meaningful, we showed how to investigate system properties using the logic CSL for probabilistic model checking. The behaviour of the systems predicted by these analysis techniques appeared generally to be in accordance with biology literature and results of others modelling approaches. Moreover, we applied the modelling approach to a real case study, the Arsenic Biosensor synthetic system developed by the Edinburgh Team for the iGEM 2006 competition. With these examples we believe to have shown that our approach, and in general the process algebra formalism, is suitable for representing fundamental mechanisms and systems of Synthetic Biology interest.

From the definition of the graphical notation to the final translation in PRISM,

all steps have been presented in a formal fashion, using pseudo-code like definitions where possible, therefore representing precise implementation guidelines for a future construction of the modelling workbench.

6.2 Future works

We list here some broad future work tracks that if accomplished can move us forward to the creation of a more complete modelling framework.

- **Implementation:** in order to test the applicability of the modelling process here presented to a wide range of scenarios, the first step is mandatory to be the actual implementation of a software workbench. The use of pseudo-code instructions in defining this work should make the construction of the graphical interface and its conversion to PEPA and PRISM quite a trivial programming task.
- **Extensions of symbols:** the symbols that we have taken from the *Registry of Standard Biological Parts* are a representative subset of the existing ones (at level of Parts abstraction). For accomplishing a complete mapping, there should be included support to other kinds of biological entities such as *ribosome binding site* parts.
- **Connection with the Registry:** the MIT hosts the *Registry of Standard Biological Parts* as a free, open and shared public database. The modelling framework software tool can be programmed to interact with the online database and therefore provide the modellers with a complete list of the available parts. The information can contain important characterisations such as quantitative data and mechanism details, as well as previously developed models.
- **Support to other process algebras:** in the ongoing work to define a better formalism for representing biological systems, several different dialects of the (stochastic) process algebra formalism have been proposed [14]. Synthetic Biology will benefit from these innovations by the application of such novel solutions to our approach, i.e. the Beta Binders support to species affinity or the compartment dynamics modelling solutions proposed in Brane Calculi.
- **Support to the abstraction hierarchy:** Synthetic Biology defines a layered structured abstraction hierarchy for facilitating the construction of synthetic sys-

tems. It would be interesting to define a formal mapping between these levels and the abstraction levels used in modelling.

6.3 Final observations

The work proposed in this thesis report is, to the best of our knowledge, the first attempt to define a specific structured relation between the in developing and emerging discipline of Synthetic Biology and the application of stochastic process algebras to biological systems modelling. We believe that this first investigation could lead to a two-fold result. Firstly, it may help in attracting the interest of biologists in adopting this formalism for systems investigations: the graphical notation that we developed permits the generation of formal models even to users not familiar with mathematical and computational languages. Although being in general a correct abstraction, the automatically generated process algebra model usually needs to be refined with respect to particular characteristics by a modeler expert. In this sense, we envisage for process algebras a role of intermediate cross-talking language between biologist and bioinformatic profiles in multidisciplinary research teams.

Secondly, with our work we hope to attract the attention of research groups working on process algebra formalism to the fertile ground represented by the Synthetic Biology discipline. In the usually poorly defined and complex field of biology, formal languages approaches to modelling find major problems in characterising entities and processes involved, as well as in finding the related quantitative information. In this context, Synthetic Biology emerges as one of the few biological fields profoundly influenced by engineering paradigms. For this reason, the application of process algebras should find a strong affinity with the increasing centrality of concepts such as standard characterised parts, compositional properties, (sub)systems modularity and well defined abstraction layers.

Appendix A

Appendix

A.0.1 PRISM Code for the Protein's Degradation example

```
ctmc
//stochastic constant for degradation
const double kDeg =0.05;
//settings for protein A
const int minA=0;
const int maxA=100;
const int initA=100;

module A
levelA:[minA..maxA] init initA;
[Deg] (levelA>minA) -> levelA : levelA'=levelA-1;
endmodule

module Rates
[Deg] (true) -> kDeg : true;
endmodule

//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
```

A.0.2 PRISM Code for the Reversible Reaction example

```
ctmc
//stochastic constant for R1
const double kR1=0.0001;
//stochastic constant for R2
const double kR2=0.1;
```

```
//settings for protein A
const int minA=0;
const int maxA=500;
const int initA=350;
//settings for protein B
const int minB=0;
const int maxB=500;
const int initB=200;
//settings for protein C
const int minC=0;
const int maxC=500;
const int initC=250;

module A
levelA:[minA..maxA] init initA;
[R1] (levelA>=minA+1) -> levelA : levelA'=levelA-1;
[R2] (levelA<=maxA-1) -> 1 : levelA'=levelA+1;
endmodule

module B
levelB:[minB..maxB] init initB;
[R1] (levelB>=minB+1) -> levelB : levelB'=levelB-1;
[R2] (levelB<=maxB-1) -> 1 : levelB'=levelB+1;
endmodule

module Ci
levelC:[minC..maxC] init initC;
[R1] (levelC<=maxC-1) -> 1 : levelC'=levelC+1;
[R2] (levelC>=minC+1) -> levelC : levelC'=levelC-1;
endmodule

module Rates
[R1] (true) -> kR1 : true;
[R2] (true) -> kR2 : true;
endmodule

//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
rewards "levelB" true : levelB; endrewards
rewards "levelC" true : levelC; endrewards
```

A.0.3 PRISM Code for the Enzymatic Reaction example

```

ctmc
//stochastic constant for R1
const double kR1=1;
//stochastic constant for R2
const double kR2=0.01;
//stochastic constant for R3
const double kR3=0.1;
//settings for protein A
const int minA=0;
const int maxA=300;
const int initA=300;
//settings for enzyme E
const int minE=0;
const int maxE=100;
const int initE=100;
//settings for complex AE
const int minAE=0;
const int maxAE=300;
const int initAE=0;
//settings for protein P
const int minP=0;
const int maxP=300;
const int initP=0;

module A
levelA:[minA..maxA] init initA;
[R1] (levelA>=minA+1) -> levelA : levelA'=levelA-1;
[R2] (levelA<=maxA-1) -> 1 : levelA'=levelA+1;
endmodule

module E
levelE:[minE..maxE] init initE;
[R1] (levelE>=minE+1) -> levelE : levelE'=levelE-1;
[R2] (levelE<=maxE-1) -> 1 : levelE'=levelE+1;
[R3] (levelE<=maxE-1) -> 1 : levelE'=levelE+1;
endmodule

module AE
levelAE:[minAE..maxAE] init initAE;
[R1] (levelAE<=maxAE-1) -> 1 : levelAE'=levelAE+1;

```

```
[R2] (levelAE>=minAE+1) -> levelAE : levelAE'=levelAE-1;
[R3] (levelAE>=minAE+1) -> levelAE : levelAE'=levelAE-1;
endmodule
```

```
module Pi
levelP:[minP..maxP] init initP;
[R3] (levelP<=maxP-1) -> 1 : levelP'=levelP+1;
endmodule
```

```
module Rates
[R1] (true) -> kR1 : true;
[R2] (true) -> kR2 : true;
[R3] (true) -> kR3 : true;
endmodule
```

```
//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
rewards "levelE" true : levelE; endrewards
rewards "levelAE" true : levelAE; endrewards
rewards "levelP" true : levelP; endrewards
```

A.0.4 PRISM Code for the Gene Expression example

```
ctmc
//stochastic constant for Deg
const double kDeg=0.01;
//stochastic constant for A expression
const double eP=0.8;
//settings for protein A
const int minA=0;
const int maxA=300;
const int initA=0;

module A
levelA:[minA..maxA] init initA;
[Deg] (levelA>minA)-> levelA : levelA'=levelA-1;
[PExp] (levelA<maxA)-> 1 : levelA'=levelA+1;
endmodule

module POp
[PExp] (true)-> eP : true;
endmodule
```

```

module Rates
[Deg] (true) -> kDeg : true;
endmodule

//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards

```

A.0.5 PRISM Code for the Gene Repression and Inducible System example

```

ctmc
//stochastic constant for Deg
const double kDeg=0.005;
//stochastic constant for Ind
const double kInd=0.00005;
//stochastic constant for A expression
const double eP=0.8;
//repressor binding rate
const double bP=0.05;
//repression rate
const double rP=0.8;
//settings for protein A
const int minA=0;
const int maxA=300;
const int initA=0;
//settings for protein Eff
const int minEff=0;
const int maxEff=300;
const int initEff=300;
//settings for protein Rep
const int minRep=0;
const int maxRep=300;
const int initRep=200;

module Eff
levelEff:[minEff..maxEff] init initEff;
[Ind] (levelEff>minEff)-> levelEff : levelEff'=levelEff-1;
endmodule

module Rep

```

```

levelRep:[minRep..maxRep] init initRep;
[Ind] (levelRep>minRep)-> levelRep : levelRep'=levelRep-1;
[P_bind_RepRep] (levelRep>0) -> levelRep : levelRep'=levelRep;
endmodule

```

```

module A
levelA:[minA..maxA] init initA;
[Deg] (levelA>minA)-> levelA : levelA'=levelA-1;
[PExp] (levelA<maxA)-> 1 : levelA'=levelA+1;
endmodule

```

```

module POp
check:[0..1] init 0;
[PExp] (check=0)-> eP : true;
[P_bind_RepRep] (check=0) -> bP : check'=1;
[P_unbind_RepRep] (check=1) -> rP : check'=0;
endmodule

```

```

module Rate
[Deg] (true) -> kDeg : true;
[Ind] (true) -> kInd : true;
endmodule

```

```

//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
rewards "levelEff" true : levelEff; endrewards
rewards "levelRep" true : levelRep; endrewards

```

A.0.6 PRISM Code for the Gene Activation and Repressible System example

```

ctmc
//stochastic constant for Deg
const double kDeg=0.005;
//stochastic constant for CoRep
const double kCoRep=0.00005;
//stochastic constant for A expression
const double eP=0.009;
//activator binding rate
const double bP=0.01;
//activation rate

```



```

const double aP=0.8;
//settings for protein A
const int minA=0;
const int maxA=300;
const int initA=0;
//settings for protein Eff
const int minEff=0;
const int maxEff=300;
const int initEff=250;
//settings for protein Act
const int minAct=0;
const int maxAct=300;
const int initAct=200;

module Eff
levelEff:[minEff..maxEff] init initEff;
[CoRep] (levelEff>minEff)-> levelEff : levelEff'=levelEff-1;
endmodule

module Act
levelAct:[minAct..maxAct] init initAct;
[CoRep] (levelAct>minAct)-> levelAct : levelAct'=levelAct-1;
[P_bind_ActAct] (levelAct>minAct) -> levelAct : levelAct'=levelAct;
endmodule

module A
levelA:[minA..maxA] init initA;
[Deg] (levelA>minA)-> levelA : levelA'=levelA-1;
[PExp] (levelA<maxA)-> 1 : levelA'=levelA+1;
endmodule

module POp
check:[0..1] init 0;
[PExp] (check=0)-> eP : true;
[P_bind_ActAct] (check=0) -> bP : check'=1;
[PExp] (check=1) -> aP : check'=0;
endmodule

module Rates
[Deg] (true) -> kDeg : true;
[CoRep] (true) -> kCoRep : true;
endmodule

```

```
//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
rewards "levelEff" true : levelEff; endrewards
rewards "levelAct" true : levelAct; endrewards
```

A.0.7 PRISM Code for the Transcription and Translation Processes example

```
ctmc
//stochastic constant for reactions
const double kDegmA=0.02;
const double kDegmB=0.02;
const double kDegA=0.01;
const double kDegB=0.01;
const double kTrasA=0.05;
const double kTrasB=0.01;
//constant for Operon transcription
const double eP=0.8;
//settings for mRNA for protein A
const int minmA=0;
const int maxmA=100;
const int initmA=0;
//settings for mRNA for protein B
const int minmB=0;
const int maxmB=100;
const int initmB=0;
//settings for protein A
const int minA=0;
const int maxA=100;
const int initA=0;
//settings for protein B
const int minB=0;
const int maxB=100;
const int initB=0;

module A
levelA:[minA..maxA] init initA;
[DegA] (levelA>minA)-> levelA : levelA'=levelA-1;
[TrasA] (levelA<maxA)-> 1 : levelA'=levelA+1;
endmodule
```

```

module B
levelB:[minB..maxB] init initB;
[DegB] (levelB>minB)-> levelB : levelB'=levelB-1;
[TrasB] (levelB<maxB)-> 1 : levelB'=levelB+1;
endmodule

module mA
levelmA:[minmA..maxmA] init initmA;
[DegmA] (levelmA>minmA)-> levelmA : levelmA'=levelmA-1;
[TrasA] (levelmA>minmA)-> levelmA : levelmA'=levelmA-1;
[PExp] (levelmA<maxmA) -> 1 : levelmA'=levelmA+1;
endmodule

module mB
levelmB:[minmB..maxmB] init initmB;
[DegmB] (levelmB>minmB)-> levelmB : levelmB'=levelmB-1;
[TrasB] (levelmB>minmB)-> levelmB : levelmB'=levelmB-1;
[PExp] (levelmB<maxmB) -> 1 : levelmB'=levelmB+1;
endmodule

module POp
[PExp] (true)-> eP : true;
endmodule

module Rates
[DegA] (true) -> kDegA : true;
[DegB] (true) -> kDegB : true;
[DegmA] (true) -> kDegmA : true;
[DegmB] (true) -> kDegmB : true;
[TrasA] (true) -> kTrasA : true;
[TrasB] (true) -> kTrasB : true;
endmodule

//reward definitions for graph plotting
rewards "levelA" true : levelA; endrewards
rewards "levelB" true : levelB; endrewards
rewards "levelmA" true : levelmA; endrewards
rewards "levelmB" true : levelmB; endrewards

```

A.0.8 PRISM Code for the Passive Transport example

```

ctmc
//stochastic constant for kExo
const double kTransp=0.005;
//settings for protein AC1
const int minAC1=0;
const int maxAC1=200;
const int initAC1=200;
//settings for protein AC2
const int minAC2=0;
const int maxAC2=200;
const int initAC2=0;

module AC1
levelAC1:[minAC1..maxAC1] init initAC1;
[Transp] (levelAC1>=minAC1+1) -> levelAC1 : levelAC1'=levelAC1-1;
endmodule

module AC2
levelAC2:[minAC2..maxAC2] init initAC2;
[Transp] (levelAC2<=maxAC2-1) -> 1 : levelAC2'=levelAC2+1;
endmodule

module Rates
[Transp] (true) -> kTransp : true;
endmodule

//reward definitions for graph plotting
rewards "levelAC1" true : levelAC1; endrewards

```

A.0.9 PRISM Code for the Vesicle Transport example

```

ctmc
//stochastic constant for kExo
const double kExo=0.1;
//stochastic constant for kPagho
const double kPagho=0.005;
//stoichiometric Vesicle
const int cV=50;
//settings for protein AC1
const int minAC1=0;

```

```

const int maxAC1=500;
const int initAC1=500;
//settings for protein AC2
const int minAC2=0;
const int maxAC2=500;
const int initAC2=0;
//settings for protein Ves
const int minVes=0;
const int maxVes=500;
const int initVes=0;

module AC1
levelAC1:[minAC1..maxAC1] init initAC1;
[Exo] (levelAC1>=minAC1+cV) -> 1 : levelAC1'=levelAC1-cV;
endmodule

module Ves
levelVes:[minVes..maxVes] init initVes;
[Exo] (levelVes<=maxVes-1) -> 1 : levelVes'=levelVes+1;
[Pagho] (levelVes>=minVes+1) -> levelVes : levelVes'=levelVes-1;
endmodule

module AC2
levelAC2:[minAC2..maxAC2] init initAC2;
[Pagho] (levelAC2<=maxAC2-cV) -> 1 : levelAC2'=levelAC2+cV;
endmodule

module Rates
[Pagho] (true) -> kPagho : true;
[Exo] (true) -> kExo : true;
endmodule

//reward definitions for graph plotting
rewards "levelAC1" true : levelAC1; endrewards
rewards "levelAC2" true : levelAC2; endrewards
rewards "levelVes" true : levelVes; endrewards

```

A.0.10 PRISM Code for the Receptor-Ligand Interactions example

```

ctmc
//stochastic constant for kProRs

```

```

const double kProRs=0.04;
//stochastic constant for kDegRs
const double kDegRs=0.01;
//stochastic constant for kAct
const double kAct=0.008;
//stochastic constant for kSig
const double kSig=0.8;
//stochastic constant for kDeAct
const double kDeAct=0.08;
//setting for Rs
const int minRs=0;
const int maxRs=50;
const int initRs=0;
//setting for L
const int minL=0;
const int maxL=20;
const int initL=20;
//setting for Ra
const int minRa=0;
const int maxRa=50;
const int initRa=4;
//setting for Rs
const int minPAct=0;
const int maxPAct=500;
const int initPAct=0;

module Rs
levelRs:[minRs..maxRs] init initRs;
[ProRs] (levelRs<maxRs) -> 1 : levelRs'=levelRs+1;
[DegRs] (levelRs>minRs) -> levelRs : levelRs'=levelRs-1;
[Act] (levelRs>minRs) -> levelRs : levelRs'=levelRs-1;
[DeAct] (levelRs<maxRs) -> 1 : levelRs'=levelRs+1;
endmodule

module L
levelL:[minL..maxL] init initL;
[Act] (levelL>minL) -> levelL : levelL'=levelL-1;
endmodule

module Ra
levelRa:[minRa..maxRa] init initRa;
[Act] (levelRa<maxRa) -> 1 : levelRa'=levelRa+1;

```

```

[Sig] (levelRa>minRa) -> kSig : levelRa'=levelRa;
[DeAct] (levelRa>minRa) -> levelRa : levelRa'=levelRa-1;
endmodule

module PAct
levelPAct:[minPAct..maxPAct] init initPAct;
[Sig] (levelPAct<maxPAct) -> 1 : levelPAct'=levelPAct+1;
endmodule

module Rates
[ProRs] (true) -> kProRs : true;
[DegRs] (true) -> kDegRs : true;
[Act] (true) -> kAct : true;
[DeAct] (true) -> kDeAct : true;
[Sig] (true) -> kSig : true;
endmodule

//reward definitions for graph plotting
rewards "levelRs" true : levelRs; endrewards
rewards "levelL" true : levelL; endrewards
rewards "levelRa" true : levelRa; endrewards
rewards "levelPAct" true : levelPAct; endrewards

```

A.0.11 PRISM Code for the Arsenic Biosensor system

```

ctmc
//stochastic constants
const double kDegArsR=0.001;
const double kDegCl=0.001;
const double eArs=0.1;
const double bArs=0.6;
const double rArs=0.2;
const double kArseniteArsR=0.005;
//stochastic constant
const double kDegUrease=0.001;
//stochastic constants
const double eCl=0.08;
const double rCl=0.02;
const double bCl=0.001;
//settings for Urease
const int minUrease=0;
const int maxUrease=150;

```

```

const int initUrease=0;
//settings for Cl
const int minCl=0;
const int maxCl=150;
const int initCl=0;
//settings for ArsR
const int minArsR=0;
const int maxArsR=150;
const int initArsR=0;
//settings for Arsenite
const int minArsenite=0;
const int maxArsenite=150;
const int initArsenite=0;

module Rates
[DegArsR] (true) -> kDegArsR : true;
[DegCl] (true) -> kDegCl : true;
[bind] (true) ->kArseniteArsR : true;
[DegUrease] (true) -> kDegUrease : true;
endmodule

module ArsOp
check:[0..1] init 0;
[ArsExp] (check=0) -> eArs : true;
[Ars_bind_ArsRRep] (check=0) -> bArs : check'=1;
[Ars_unbind_ArsRRep] (check=1) -> rArs : check'=0;
endmodule

module Cl
levelCl:[minCl..maxCl] init initCl;
[DegCl] (levelCl>minCl) -> levelCl : levelCl'=levelCl-1;
[ArsExp] (levelCl<maxCl) -> 1 : levelCl'=levelCl+1;
[Cl_bind_ClRRep] (levelCl>0) -> levelCl : true;
endmodule

module ArsR
levelArsR:[minArsR..maxArsR] init initArsR;
[DegArsR] (levelArsR>minArsR) -> levelArsR : levelArsR'=levelArsR-1;
[ArsExp] (levelArsR<maxArsR) -> 1 : levelArsR'=levelArsR+1;
[Ars_bind_ArsRRep] (levelArsR>0) -> levelArsR : true;
[bind] (levelArsR>minArsR) -> levelArsR : levelArsR'=levelArsR-1;
endmodule

```



```
module Arsenite
levelArsenite:[minArsenite..maxArsenite] init initArsenite;
[ArseniteArsR] (levelArsenite>minArsenite) -> levelArsenite : true;
endmodule

module Urease
levelUrease:[minUrease..maxUrease] init initUrease;
[DegUrease] (levelUrease>minUrease) -> levelUrease : levelUrease'=levelUrease-1;
[ClExp] (levelUrease<maxUrease) -> 1 : levelUrease'=levelUrease+1;
endmodule

module ClOp
check1:[0..1] init 0;
[ClExp] (check1=0) -> eCl : true;
[Cl_bind_ClRRep] (check1=0) -> bCl : check1'=1;
[Cl_unbind_ClRep] (check1=1) -> rCl : check1'=0;
endmodule

//reward definitions for graph plotting
rewards "levelArsR" true : levelArsR; endrewards
rewards "levelCl" true : levelCl; endrewards
rewards "levelArsenite" true : levelArsenite; endrewards
rewards "levelUrease" true : levelUrease; endrewards
```


Bibliography

- [1] Steven A. Benner and Michael A. Sismour. Synthetic biology. *Nat Rev Genet*, 6(7):533–543, July 2005.
- [2] William H. Elliott and Daphne C. Elliott. *Biochemistry and Molecular Biology*. Oxford University Press, USA, January 2005.
- [3] Drew Endy. Foundations for engineering biology. *Nature*, 438(7067):449–453.
- [4] Drew Endy and Roger Brent. Modelling cellular behaviour. *Nature*, 409(6818):391–395, January 2001.
- [5] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, November 2002.
- [6] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pac Symp Biocomput*, pages 459–470, 2001.
- [7] 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, October 2001.
- [8] Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.
- [9] T.F Knight. Plasmid distribution 1.00 of standard biobrick components. *MIT Synthetic Biology Working Group Reports*.
- [10] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, January 2000.
- [11] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–342, January 2000.

- [12] J. Brown. The igem competition: building with biology. *Synthetic Biology*, 1:3–6, 2007.
- [13] A. Regev and E. Shapiro. Cellular abstractions: Cells as computation. *Nature*, 419(6905):343, 2002.
- [14] M. L. Guerriero, D. Prandi, C. Priami, and P. Quaglia. Process calculi abstractions for biology. In *Technical Report at The Microsoft Research - University of Trento, Centre for Computational and Systems Biology*, number 13, 2006.
- [15] D. Chiarugi, Michele Curti, Pierpaolo Degano, and Roberto Marangoni. Vice: A virtual cell. In *CMSB*, pages 207–220, 2004.
- [16] Ralf Blossey, Luca Cardelli, and Andrew Phillips. A compositional approach to the stochastic dynamics of gene networks. *Transactions in Computational Systems Biology*, 3939:99–122, 2006.
- [17] Luca Cardelli. A compositional approach to the stochastic dynamics of gene networks. *CONCUR 2005 Concurrency Theory*, pages 4–4, 2005.
- [18] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.
- [19] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.
- [20] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
- [21] S. Gilmore and J. Hillston. The PEPA Workbench: A Tool to Support a Process Algebra-based Approach to Performance Modelling. In *Proceedings of the Seventh International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, number 794 in *Lecture Notes in Computer Science*, pages 353–368, Vienna, May 1994. Springer-Verlag.
- [22] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

- [23] M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220–270. Springer, 2007.
- [24] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.
- [25] S.L. Moodie, A.A. Sorokin, I. Goryanin, and P. Ghazal. A graphical notation to describe the logical interactions of biological pathways. *Journal of Integrative Bioinformatics*, 3(2):36, 2006.
- [26] H. Kitano. A graphical notation for biochemical networks. *Biosilico*, 1:169, 2003.
- [27] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. *Systems Biology in Practice: Concepts, Implementation and Application*. Wiley-VCH, May 2005.
- [28] David L. Nelson and Michael M. Cox. *Lehninger Principles of Biochemistry, Fourth Edition*. W. H. Freeman, April 2004.
- [29] N. Joshi et al. Use of synthetic biology to generate a novel arsenic biosensor. *Internal Technical Report at University of Edinburgh*, 2006.
- [30] Muffy Calder, Stephen Gilmore, and Jane Hillston. Modelling the influence of RKIP on the ERK signalling pathway using the stochastic process algebra PEPA. In Anna Ingolfssdottir and Hanne Riis Nielson, editors, *Proceedings of the Bio-Concur Workshop on Concurrent Models in Molecular Biology*, London, England, August 2004.
- [31] Muffy Calder, Adam Duguid, Stephen Gilmore, and Jane Hillston. Stronger computational modelling of signalling pathways using both continuous and discrete-state methods. In Corrado Priami, editor, *Proceedings of the Fourth International Conference on Computational Methods in Systems Biology (CMSB 2006)*, volume 4210 of *Lecture Notes in Computer Science*, pages 63–77, Trento, Italy, 2006. Springer.