PhD Dissertation

International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Modelling and Inference strategies for Biological Systems

Alida Palmisano

Advisor:

Prof. Corrado Priami

Università degli Studi di Trento

Co-Advisor:

Dott. Paola Lecca

The Microsoft Research - University of Trento

Centre for Computational Systems Biology

December 2010

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to", said the Cat.
"I don't much care where –" said Alice.
"Then it doesn't matter which way you go", said the Cat
"–so long as I get somewhere" , Alice added as an explanation.
"Oh, you're sure to do that" , said the Cat, *"if you only walk long enough."*

[Charles Lutwidge Dodgson]

Abstract

For many years, computers have played an important role in helping scientists to store, manipulate, and analyze data coming from many different disciplines. In recent years, however, new technological capabilities and new ways of thinking about the usefulness of computer science is extending the reach of computers from simple analysis of collected data to hypothesis generation.

The aim of this work is to provide a contribution in the Computational Systems Biology field. The main purpose of this recent discipline is to enhance the intertwined relationship connecting Biology and Computer Science, by developing tools and theoretical frameworks able to formally and quantitatively investigate the interactions among the components of biological systems. The final goal of these efforts is to assemble the different pieces into a working model of a living, responding, reproducing cell; a model that can be used for performing in-silico tests and simulations in order to understand and predict possible emergent properties.

In this thesis we present the application to real biological case studies of a specific concurrent modelling language (derived by the metaphors of "molecules-as-object" –introduced by Fontana– and "cells-as-computations" –introduced by Regev and Shapiro– at the end of last century) and the development and implementation of a tool for inferring knowledge from experimental data in order to link the numerical aspects of a model to real wet-lab data.

Keywords

[systems biology, stochastic modelling, process calculi, kinetic inference, cell cycle]

Contents

1 Introduction **1**

1.1 The context 1
1.2 The problem 4
1.3 The contribution 6
1.4 Structure of the Thesis 8
1.5 Publications 8

2 Preliminaries **11**

2.1 Modelling of Biological Systems 11
2.2 Inference of Biological Systems 17
2.3 Cell cycle models 22

I The modelling framework **29**

3 Modelling with BlenX **31**

3.1 The language 31
3.2 Translation from ODEs to BlenX 38
3.3 Issue: different abstraction levels 41

4 Budding Yeast Cell Cycle **45**

4.1 Introduction 45
4.1.1 Physiology of the cell cycle 46
4.1.2 Molecular mechanism of cell cycle control 48
4.1.3 *Saccharomyces cerevisiae* regulatory network 51
4.2 Irreversibility study 53
4.3 Viability study 59
4.4 Stochastic analysis 64
4.5 Stochastic pedigree analysis 74

4.6 Preliminary conclusions . . . 81

II The inference framework **85**

5 KInfer **87**
5.1 The model for inference . . . 88
5.1.1 Variance of the estimated parameters . . . 91
5.1.2 Parameter space restriction . . . 92
5.1.3 Parameter inference in not - mass action models . . . 94
5.2 The inference tool: KInfer . . . 96
5.3 Some concluding considerations . . . 101

6 Case studies **103**
6.1 Gene transcriptional regulation . . . 103
6.2 Didactic biochemical network . . . 106
6.3 Genetic regulatory network . . . 107
6.4 Real case studies . . . 109
6.4.1 Glucose metabolisms of Lactococcus lactis . . . 110
6.4.2 Binding affinity of IκB kinase . . . 112
6.4.3 A transcription control of cell cycle . . . 115
6.5 Other existing inference tools . . . 120

7 Discussions and Future work **123**

8 Conclusions **129**

Bibliography **137**

A Appendix **153**
A.1 BlenX model of Budding Yeast Cell Cycle . . . 153
A.1.1 .prog file . . . 153
A.1.2 .types file . . . 158
A.1.3 .func file . . . 158
A.2 Stochastic Simulation Algorithm . . . 163
A.3 Continuous Stochastic Logic and PRISM . . . 164
A.4 Deterministic data (Chen model) . . . 165

List of Tables

2.1 A concise picture of the mapping between biology and process calculi . . . 13

3.1 BlenX input files of the Budding Yeast Cell Cycle (running example) 36

4.1 Stochastic characterization of all the available mutants described in the deterministic framework 69

6.1 Estimated parameter values: gene transcriptional regulation (model 1) . . 104
6.2 Estimated parameter values: gene transcriptional regulation (model 2) . . 105
6.3 Estimated parameter values: didactical biochemical network 106
6.5 Estimated parameter values: gene regulatory network 109
6.6 Estimated parameter values: regulation of glycolysis in L. lactis 110
6.7 Estimated parameter values: binding affinity of IκB kinase 114
6.9 Estimated parameter values: Cdc20 transcription sub-network 117
6.11 Estimates of the parameters of the Gauss error function approximating the Hill function 119

List of Figures

1.1 Modelling cycle of a biological system 5

2.1 Structure of a cell 23
2.2 Budding yeast cells. 24

3.1 Graphical representation of a BlenX entity. 32
3.2 Intuitive behavior of some BlenX primitives. 34
3.3 Model, used as a running example for explaining the translation procedure, of the antagonism between Cdk/CycB complex and the Cdh1/APC that drives cell cycle oscillations. 35
3.4 Example of the inverse problem causing inconsistencies between the deterministic and the stochastic behaviour of a chemical reaction system. 40
3.5 Inconsistencies between different ways of unpacking the same complex regulatory mechinism. 43

4.1 Phases of the cell cycle. The evolution in time of the budding yeast case is shown. 47
4.2 A seesaw metaphor for the antagonism between Cdk/Cyclin and Cdh1/APC and Sic1 proteins that drive the cell cycle oscillations 49
4.3 Protein waves during the cell cycle 50
4.4 Regulatory network of cell cycle 51
4.5 Network of the antagonistic interactions between Cdk/CycB and Cdh1/APC 54
4.6 Visual representation of noise-free oscillations of Cdk/CycB and Cdh1/APC 56
4.7 Probability plots of the behavior of Cdk/CycB and Cdh1/APC with respect to the value of a parameter of the cell cycle model 57
4.8 Sample simulations of the wild type model 58
4.9 Probability of a single Cdh1/APC peak between two cell divisions 59
4.10 Graphical representation of cell cycle engine 60
4.11 Equations for the activation/inactivation of the Cdc20 protein and the rate law for the growing of the mass 60

4.12 BlenX code resulting from the translation of a subset of mathematical equations . . . 61
4.13 Statistics on properties of a simplified model of cell cycle . . . 62
4.14 Stochastic simulations for a nutritional sensitive mutant . . . 63
4.15 BlenX code with templates . . . 64
4.16 Rules that define stages of the cell cycle and arrest/error type . . . 66
4.17 Statistics on properties of a complete model of cell cycle . . . 67
4.18 For all the viable mutants, a comparison of the statistics collected from the deterministic solution of the model and stochastic runs of the same parameter sets . . . 68
4.19 For all the inviable mutants, a comparison of the arrest/error type deduced from the deterministic solution of the model and stochastic runs of the same parameter sets . . . 72
4.20 Example of an oscillating but inviable mutant . . . 73
4.21 Graphical representation of the idea behind spPeAn^{BY}. . . . 75
4.22 Graphical representation of spPeAn^{BY}architecture. . . . 76
4.23 Pedigree analysis of the stochastic version of the Chen model . . . 78
4.24 Pedigree tree with dead branches . . . 78
4.25 Statistics on the output of the pedigree analysis of the stochastic version of the Chen model . . . 79
4.26 Histograms of the genealogical classification of a colony of budding yeast cells . . . 80
4.27 Histograms of the synchronicity of a colony of budding yeast cells . . . 80

5.1 Screenshots of the inference tool: KInfer . . . 96
5.2 Graphical representation of the general architecture of KInfer . . . 97

6.1 Gene transcriptional regulation (model 1) . . . 104
6.2 Time series of gene transcription case study (model 1) . . . 104
6.3 Gene transcriptional regulation (model 2) . . . 105
6.4 Time series of gene transcription case study (model 2) . . . 105
6.5 Didactical example of a biochemical network . . . 106
6.6 Time series of the didactical biochemical network case study . . . 107
6.7 Model of a gene regulatory network . . . 108
6.8 Time series of the gene regulatory network model . . . 109
6.9 Model of the pathway of regulation of glycolysis in L. lactis . . . 111
6.10 Time series of the pathway of regulation of glycolysis in L. lactis . . . 112
6.11 Model of the binding affinity of IκB kinase . . . 113

6.13 Time series of GST-IκB-P 114
6.14 A simplified model for the transcription of the gene Cdc20 in budding yeast 115
6.16 Time course of the Cdc20 mRNA in a simplified model for the transcription of the Cdc20 gene 118
6.17 Solutions of the model including Cdc20 transcription process with the rate coefficients inferred from the different experimental time courses 118
6.18 Fit of Gauss error function to Hill function 120

Chapter 1

Introduction

1.1 The context

For many years, computers have played an important role in helping scientists to store, manipulate, and analyze data coming from many different disciplines. In recent years, however, new technological capabilities and new ways of thinking about the usefulness of computer science is extending the reach of computers from simple analysis of collected data to hypothesis generation. The authors of [62] predict that within a decade, powerful tools will enable automated, high-volume hypothesis generation to guide high-throughput experiments in biomedicine, chemistry, physics, and even social sciences. In order to go in this direction, a shift from the old way of studying the different scientific fields in isolation was needed and already started: the cross-fertilization between sciences like biology, computer science, sociology, ecology, economy, mathematics, physics, etc. is critical to advance human knowledge and allows the discovery of many unexpected connections between seemingly disjoint fields of science.

For this thesis work, we decided to focus on the intertwined relationship that, in the last few years, has involved biology and computer science.

In the last thirty years, the constant advances in technologies and experimental techniques in molecular biology have led to a massive increase in the quantity of data scientists produce: in this respect, computer science has helped life-scientists to manage this enormous amount of knowledge. The result of this effort is what we call today *bioinformatics*. This first convergence between computing and biology led scientists to a deeper comprehension of the basic mechanisms governing living organisms, leading to unthinkable achievements like the *Human Genome Project* that have begun of the *post-genomic* era.

However, the initial idea behind the bioinformatic approach relies on the reductionist concept that a complex system is nothing but the sum of its parts, so a biological system

can be understood by studying its individual components (e.g. molecules and proteins) and the connections among them [114]. The method is based on the assumption that the single components contain enough information to explain the complexity of the system as a whole. Although reductionist methods form the basis for many of the well-developed areas of modern science (including physics, chemistry and cell biology), the limit of reductionism's usefulness stems from emergent properties of complex systems, which are more common at certain levels of organization.

As Kitano points out in [105] a complete system-level understanding requires a shift in the notion of *what to look for* in biology. Also the Noble laureate Paul Nurse [144] writes that a proper understanding of the complex regulatory networks making up cellular systems like the cell cycle will require a shift from common sense thinking: "We might need to move into a strange more abstract world more readily analyzable in terms of mathematics". So, while the understanding of a molecular network at the level of single genes and proteins continues to be important, more effort is needed to focus on understanding the system's structure and dynamics at a higher level of abstraction. These concepts lead to the establishment of the new paradigm that in the last years has been identified with *systems biology*.

This new approach has its root in the use of modelling and simulation, combined with experiment, to explore network behavior in biological systems – in particular their dynamic nature. To give a precise meaning to the term of "systems biology" is difficult, here just a few examples of definitions are listed:

> To understand complex biological systems requires the integration of experimental and computational research – in other words a systems biology approach [105]

> [...] the objective of systems biology [can be] defined as the understanding of network behaviour, and in particular their dynamic aspects, which requires the utilization of mathematical modeling tightly linked to experiment. [27]

> By discovering how function arises in dynamic interactions, systems biology addresses the missing links between molecules and physiology. Top-down systems biology identifies molecular interaction networks on the basis of correlated molecular behavior observed in genome-wide omics studies. Bottom-up systems biology examines the mechanisms through which functional properties arise in the interactions of known components. [16]

One of the main reasons why it is so difficult to come up with a concise definition of systems biology might be that every definition needs to respect a delicate balance between

different and complementary aspects of the discipline: the integration of experimental and computational approaches, the balance between genome-wide systematic approaches and smaller-scale quantitative studies, top-down versus bottom-up strategies to solve systems architecture and functional properties. But despite the diversity in opinions and views, two main aspects are conserved across these definitions: a) a system-level approach attempts to consider *all the components* of a system; b) the interactions of the components are translated to their functions by a *computational model*.

It is clear how systems biology is strongly inter-disciplinary and calls for the integration and convergence of many different sciences. Its domain indeed spreads from several branches of biology to more distant fields of study, such as physics, mathematics, statistics and informatics. The main role of these latter distant disciplines is to devise adequate formal tools and concepts to describe and model efficiently biological systems, analyse their properties, and reproduce and predict their behavior through computer-based simulations (e.g., [184, 115, 124, 129, 37]), with the ultimate and challenging goal of delineating the basis for a foundation theory of systems biology.

Although advances in accurate, quantitative experimental approaches will doubtless continue, insights into the functioning of biological systems will not result from purely intuitive assaults, because of the intrinsic complexity of biological systems [105]. So a combination of experimental and computational techniques is expected to resolve this problem. Recently researchers working in different fields of life-sciences have expressed the need for systematic approaches and computer models of biochemical and signalling networks in order to arrive at testable quantitative predictions despite the complexity of these networks [106]. For example, Hartwell and colleagues [85] argue that the best test of our understanding of cells will be to make quantitative predictions about their behavior and verify them in the lab: this will require detailed simulations of the biochemical process taking place within cells. This procedure requires that results of computer-executable models are first compared with experimental observation: inconsistencies found at this stage means that the assumptions behind our knowledge of the system are at best incomplete. Models that pass this initial validation can then be used to make predictions to be tested by experiments, as well as to explore questions that are not amenable to experimental inquiry. Although traditional bioinformatics has been used widely for genome analysis, simulation-based approaches have received little mainstream attention. This is now changing. Current experimental molecular biology is now producing the high-throughput quantitative data needed to support simulation-based research.

However, in order to have useful simulation results, it is not enough to have good data from which to infer reliable numeric information to feed computer-executable models, it is also very important to choose a suitable formalism to write the models because different

formalisms allow different kinds of analysis, i.e. different kinds of question that we are able to answer with respect to that system: citing Nurse [145], "we require the development of the appropriate languages to describe information processing in biological systems".

In recent years there has been increasing interest in the application of process calculi (a formalism derived from theoretical computer science) in the modelling and analysis of biological systems [68, 161, 25, 67, 154, 37]. Process algebras have some interesting properties that make them particularly useful in this context. First of all, they offer compositionality, i.e. the possibility of defining the whole system starting from the definition of its subcomponents. Secondly, process algebras give a formal representation of the system avoiding ambiguity. Thirdly, the different parts of a biological system can be modelled as concurrent independent sub-systems running in parallel in an asynchronous way (with the possibility of adding synchronization just when it is needed): this approach is in contrast with the forced synchronicity of the mathematical formalism of differential equations implied by the contemporaneous solution of the set of equations defining the whole system. The application of independent rules for looking at the dynamical evolution of the system is a more appropriate representation of what is actually going on in biological systems.

Motivated by the intention to see those concepts of computational modelling and inference of biological systems applied on real case studies and to give a contribution in this field we started three years ago by exploring the usage of some specific computational tools and formalisms that support different stages of the modelling cycle of systems biology.

1.2 The problem

The physiological characteristics of a cell are determined by networks of interacting proteins that process energy, material and information. Confined to a few picoliter of cytoplasm these processing and control systems are not only as complex as a Boeing 777 but are also able to make exact replicas of themselves from CO_2, NO_3^-, PO_4^{3-} and a drop of mineral water [188]. We would like to know how these complex machines work, but they do not come with instruction manuals or schematic wiring diagrams. It is the grand challenge for post-genomic life scientists to deduce diagrams and write manuals. This effort will take a variety of resources and approaches: genetics and biochemistry, hardware and software, high-throughput and low-throughput technologies, hypothesis-driven and discovery-driven experiments, etc. The grand challenge of post-genomic cell biology is to assemble these pieces into a working model of a living, responding, reproducing cell; a model that gives reliable account of how the physiological properties of a cell derive from its underlying molecular machinery.

In all sciences, models are used because they represent in an accessible, abbreviated and convenient form the more complex and detailed reality [197]. Models can serve as explanatory tools for summarizing the state of knowledge, and can act as objects of predictive experiments. Most importantly, a model is a representation of some reality that embodies essential and interesting aspects of that reality. Mathematical and computational models help biologists to better understand cell physiology by unraveling the underlying dynamical behavior of the system and by allowing a deep investigation of complex interactions of regulatory molecules when different – and possibly contradictory – hypotheses can fit a specific biological scenario.

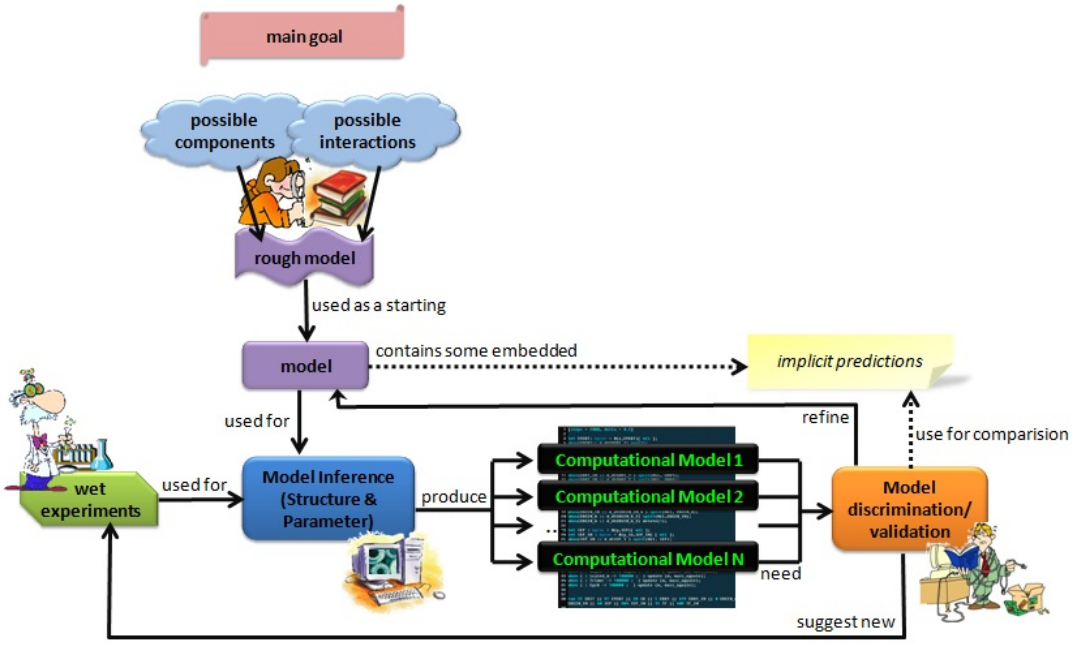


Figure 1.1: Modelling cycle of a biological system

The modelling cycle that computational biologists usually follow is the one depicted in Fig. 1.1. In the context of cellular modelling, for example, one of the *main goals* is to gain insight into the molecular interactions that are responsible for the behavior of the cell. To do so, a quantitative model of the cell must be developed and the development of such a model is iterative. It begins with a *rough model* of the cell, based on some *knowledge of its components* and *possible interactions* among them. Although the assumptions underlying the model are insufficient and may even be inappropriate for the system being investigated, this rough model provides some initial hypotheses about the structure of the interactions that govern the system and, most importantly, it contains, by the way in which it has been constructed and the language that has been chosen, *implicit predictions* about the system's response under different kinds of perturbation. Those perturbations

are fundamental for proceeding in the modelling cycle, because they drive the *model discrimination/validation* step that consists of the comparison of the models predictions to measurements taken in the different perturbed conditions. This comparison indicates where and how the model must be *refined* in order to match the measurements more closely and/or it can *suggest new experiments* that can restart the modelling cycle from the beginning.

Because of the huge amount of information, knowledge and studies that today are available to scientists, it is crucial that individual research groups are able to exchange their models, their ideas and their methodologies in software environments that are available to all. One of the obstacles that gets in the way of pursuing this goal is the lack of a global framework thought to assist and follow every step of the modelling cycle of a biological system, i.e. a set of inter-operable tools that can be used in isolation in order to fulfill specific tasks or as a whole in order to guide the work of a biologist. So each step of the modelling cycle explained above needs to be supported by automatic tools and formal languages able to guide biologists in the refinement of his/her knowledge of the investigated system: in other words, each of the steps above is an interesting open challenge from the computer science point of view.

In conclusion, developing novel quantitative conceptual and computational tools smoothly connecting models and experiments can give to life scientists a deeper understanding of fundamental biological principles. This approach can enable scientists studying a particular natural system, such as a biochemical pathway, to identify and fill in missing pieces, and traverse reasoning chains much longer than with the unaided mind.

1.3 The contribution

The contribution of this thesis is the application to real biological case studies of a specific computational modelling language and the development of a tool for inferring knowledge from experimental data in order to link the numerical aspect of the model to real wet-lab data.

In particular we chose to use the BlenX language (a recently developed stochastic process algebra) to model a well-known biological process: the cell cycle of budding yeast. In our thesis, we will present a general procedure that can be used to translate rate equation models into the chosen computational framework in an easy way. This same procedure can be applied to any existing ordinary differential equation model for which a more detailed stochastic characterization holds the potential to reveal aspects and behaviors that cannot be captured with a deterministic approach. We used different BlenX models built with this proceduce to quantify the strength of the irreversibility of

the "Start" transition of the budding yeast cell cycle with respect to a specific parameter in a simple core model; to understand the role played by the noise in peculiar mutants at the border of life and death; and finally to analyze the asymmetrical growing and division in the most complete model of the budding yeast cell cycle existing in the literature.

The motivation behind the choice of the BlenX language is three-fold: firstly it is supported by a set of tools that allow us to perform stochastic simulations of a system; secondly it is a language designed for modelling biological systems at different levels of abstraction. This means that the language directly provides primitives that enable the user to put in the same model processes described with all their elementary steps and processes that are abstracted by a complex mechanism whose precise details are not known (or not relevant) and only the overall shape of its outcome is needed for the sake of the current study. A side effect of the presence of the primitives for including abstracted behaviors in a model, is that a translation of existing mathematical models of biological systems is almost straightforward (and will be presented in the following chapters): this is a fundamental merit of a language because in the literature many complex models have already been constructed and characterized in detail, and the knowledge contained in those validated model needs to be transferred to models written with different languages as easily as possible in order to allow the discovery of further incremental insights about the system. The third main advantage of chosing BlenX as a modelling language with respect to mathematical formalisms lies in the fact that the compositional and modular nature of process algebra's description of the system makes it easy, for example, to implement a change in the hypothesised role of a reagent within a network or to integrate two sub-models into a bigger one: in general those modifications will involve changing only the code representing the behavior of the specific reagent or of the species interconnecting the two sub-systems, whereas the impact on the ODEs may be pervasive.

A modelling language like BlenX is useful for defining the structure of the model, but in order to be able to use the model to perform stochastic simulations of the system, quantitative information needs to be included in the model's definition. Often, initial conditions can be determined experimentally, but the kinetic rate constants governing the dynamical behavior of the system in time are frequently not accessible directly through experiments. Therefore methods that estimate rate constants with the maximum precision and accuracy are needed. We present a new method for estimating rate coefficients from noisy observations of concentration levels at discrete time points. This is traditionally done by fitting procedures (e.g. by the least-squares estimator) or by computationally intense methods like Bayesian approaches. We propose an alternative technique based on a probabilistic, generative model of the variations in reactant concentration. Our method returns the rate coefficients, the level of noise and an error range on the estimates of rate

constants. Its probabilistic formulation is key to a principled handling of the noise inherent in biological data, and it allows for a number of further extensions. The mathematical procedure presented here has been implemented in a software tool, named KInfer.

1.4 Structure of the Thesis

- **Chapter 1** is this introduction;
- **Chapter 2** gives basic preliminaries about the state of the art on the main biological case study chosen (i.e. the budding yeast cell cycle), process calculi (with an overview of the main ones applied or developed for systems biology) and inference methods for chemical reaction networks;
- **Chapter 3** describes some basic strategies to build a BlenX model and the straightforward translation procedure of an ordinary differential equation model into this process algebra framework;
- **Chapter 4** shows detailed studies of many models of the budding yeast cell cycle, obtained using the translation procedure described in the previous chapter. The models are of different sizes and levels of abstraction and each of them allows a different kind of analysis (from irreversibility studies, to statistical characterization of peculiar properties, to pedigree analysis);
- **Chapter 5** presents the mathematical inference framework and the tool that we developed on top of it;
- **Chapter 6** shows the application of KInfer (the tool presented in the previous chapter) to both synthetic and real biological case studies;
- **Chapter 7** discusses the limitations of the current implementation of the framewoks and some interesting possible future developments of the different techniques;
- **Chapter 8** draws some conclusions and summarises the thesis work.

1.5 Publications related to the work presented in this thesis

Book Chapters

A. Csikász-Nagy, A. Palmisano, J. Zamborszky, **Molecular network dynamics of cell cycle control: transitions to *Start* and *Finish***, *in Methods in Molecular Biology (Series editor: John M. Walker)*, 2011, Humana Press (Springer), To appear

P. Lecca, A. Palmisano, **The present and the future perspectives of biological network inference**, *in Bioinformatics and Computational Systems Biology: Recent Advances and Applications*, 2011, IGI-global (ed), To appear

A. Palmisano, C. Priami, **Cell cycle modeling, process algebra**, *in Encyclopedia of Systems Biology*, 2010, Springer (ed), To appear

L. Dematte', R. Larcher, A. Palmisano, C. Priami, A. Romanel, **Programming Biology in BlenX** , *in Systems Biology for Signaling Networks*, 1: 777-821, Ed. Sangdun Choi, Springer, 2010, DOI: 10.1007/978-1-4419-5797-9

International journals

P. Ballarini, T. Mazza, A. Palmisano, A. Csikász-Nagy, **Studying irreversible transitions in a model of cell cycle regulation**, 2009, Electronic Notes in Theoretical Computer Science, 232, pp.39-53, DOI 10.1016/j.entcs.2009.02.049

P. Lecca, A. Palmisano, A. E. Ihekwaba, C. Priami, **Calibration of dynamic models of biological systems with KInfer**, *European Biophysics Journal*, 2009, Springer Berlin/Heidelberg

P. Lecca, A. Palmisano, A. E. Ihekwaba, **Inferring the kinetic constants of cyclin-triggered expression of Cdc20 gene in eukaryotic cell cycle** Online Journal of Bioinformatics, 11(2):177-216, May, 2010

International conferences and workshops

P. Lecca, A. Palmisano, **Identification of biochemical chaotic systems with KInfer**, *Int. Journal of Simulation: Systems, Science & Technology*, United Kingdom Simulation Society, To Appear

P. Lecca, A. Palmisano, A. E. Ihekwaba, **Correlation-based network inference and modelling in systems biology: the NF-κB signalling network case study**, 2010, *Proc. Int. Conf. on Intelligent Systems, Modelling and Simulation (ISMS2010)*, IEEE

Computer Society, pp. 170-175

A. Palmisano, **Coding biological systems in a stochastic framework: the case study of budding yeast cell cycle**, 2010, *Proceedings of 1st International Conference on Bioinformatics (Bioinformatics 2010)*

P. Lecca, A. Palmisano, C. Priami, **Deducing chemical reaction rate constants and their regions of confidence from noisy measurements of time series of concentration**, 2009, Proc. of the 11th Int. Conference on Computer Modelling and Simulation (UKSim 2009)

P. Lecca, A. Palmisano, C. Priami, G. Sanguinetti, **A new probabilistic generative model of parameter inference in biochemical networks**, 2009, Proc. of the 2009 ACM Symposium on Applied Computing

A. Palmisano, I. Mura, C. Priami, **From ODEs to Language-based, executable models of Biological Systems**, 2009, Proc. of Pacific Symposium on Biocomputing 2009 (PSB 2009)

P. Lecca, A. Palmisano, **On the parameter inference in chaotic chemical systems**, 2009, *Proc. of European Symposium on Computer Modeling and Simulation (UKSim 2009)*, pp. 175-180, IEEE Computer Society

P. Lecca, A. Palmisano, **Estimating the parameters of cyclin-triggered gene expression in cell cycle control network**, 2009, *Proc. of Int. Conference on Computational Intelligence, Modelling and Simulation(ISMS2009)*, IEEE Computer Society, pp. 164-169

P. Lecca, G. Sanguinetti, A. Palmisano, C. Priami, **A new method for inferring rate coefficients from experimental time-consecutive measurements of reactant concentrations**, 2007, Proc. of the 8th International Conference on Systems Biology (ICSB 2007)

Chapter 2

Preliminaries

2.1 Modelling of Biological Systems

What is a model? In the broadest sense, a model is an abstract representation of objects or processes that explains features of these objects or processes that are interesting from the point of view of what the modeller wants to observe. For instance, the strings composed of the letters A, C, G, and T are used as a model for DNA sequences. In many cases when we consider modelling biological systems the first model that is built is a cartoon of a reaction network showing different shapes for proteins and arrows for reactions between them; then this informal description of the system is translated into a formal framework in order to be able to study its qualitative/quantitative characteristics and its dynamical properties. Model assignment is not unique and modelling is a subjective and selective procedure: the choice of a mathematical model or an algorithm depends on the problem, the purpose and the intention of the investigator: different models, using different formalisms, may highlight different aspects of the same instance providing different insights into the system. An important disadvantage of this ambiguity is that the diversity of modelling approaches makes it difficult to merge established models (e.g. for individual metabolic pathways) into larger super-models (e.g. for the complete cellular metabolism). However, the main advantage (and purpose) of a computational model of any sort, is that once it has been constructed, effects of possible perturbations can be predicted fairly cheaply in silico, even simulating conditions that are not easily accessible with experiments in wet-lab.

A wide variety of cellular models have been proposed, each of different complexity and abstraction. For example, chemical kinetic models attempt to represent a cellular process as a system of distinct chemical reactions. In this case the network state is defined by the instantaneous quantity (or concentration) of each molecular species of interest and molec-

ular species may interact via one or more reactions. Traditionally, chemical kinetics have been analyzed using ordinary differential equations (ODE for short): the use of differential equations for describing biochemical processes makes certain assumptions that are not always justified [106]. One assumption is that variables can attain continuous values. This is obviously a simplification since the underlying biological objects (i.e. molecules) have a discrete nature. As long as molecule numbers are sufficiently large this is no problem, but if the involved molecule numbers are only on the order of dozens or hundreds, the discreteness should be taken into account. Another important assumption of ordinary differential equations is that they treat the described process as deterministic. Random fluctuations are normally not part of differential equations. Again, this presumption does not hold for very small systems. Moreover, ODEs are an aggregate description of the system: if a system can be described by a set of fluxes acting on the different entities (i.e. on the system variables), then ODEs hide part of the logical structure of such fluxes by combining them into equations.

A solution to the limitations described above is to use a stochastic simulation approach that explicitly models the interactions between entities and calculates the change of the number of molecules of the participating species during the time course of a chemical reaction. Even if stochasticity can be added to the deterministic framework using the formalism of stochastic differential equations (which consist of an ordinary differential equation containing a deterministic part and an additional noise term [113]), believing in the idea expressed by Nurse in Nature [145] that "life science field requires the development of new and more appropriate languages to describe biological systems" and inspired by the metaphors of *molecules-as-object* introduced by Fontana [68] and the abstraction of *cells-as-computation* suggested by Regev and Shapiro [161] in their seminal works, we decided to use a recently developed process algebra language (called BlenX) to model a specific biological system, the cell cycle of budding yeast. In the rest of this section, in order to contextualize our study, we are going to present the state of the art of process algebra languages already used for modelling biological systems.

2.1.1 Process algebra for modelling biological systems

The main advantage of a computational model versus a mathematical one is that the former is executable and not just simply solvable [67]. Execution means that we can predict/describe the flow of control between molecules and reactions (e.g., not only the time, but also the causality relation among the events that constitute the history of the dynamics of the model). In other words, the computational modelling interpretation is similar to programming the step-by-step behavior of a system, rather than describing only its outcome with respect to time [154]. Various computational approaches have been

Biology	Process Calculi
Entity	Process
Interaction capability	Communication channel name
Interaction	Synchronization/Communication
Modification/Evolution	State change

Table 2.1: A concise picture of the mapping between biology and process calculi. In the process calculi interpretation, a biological entity (e.g. a protein) is seen as a computation unit (i.e. a process) with interaction capabilities abstracted as communication channel names. Similarly to biological entities, which interact/react through complementary capabilities, processes synchronize/communicate executing, on communication channels, complementary send/receive actions. The modifications/evolutions of molecules after reactions are represented by state changes following communications.

proposed and equipped with supporting software tools (e.g., boolean networks [103], Petri nets [90], Bayesian networks [70], graphical gaussian models [139], process calculi [156], rule-based modelling [93]).

In their seminal works, Fontana and Buss [68] and Regev and Shapiro [161] suggested to abstract *molecules-as-objects* or *cells-as-computation* and proposed that languages formerly used in the study of networks of computers could be usefully employed and extended to model biological processes (see Table 2.1).

The abstraction of molecules as parallel, interacting computational entities opened the unexpected application of the realm of concurrency theory to biology. As a consequence many modelling languages have been adapted or newly developed for building biological models and performing stochastic simulations (i.e. Stochastic π-calculus [156, 153], BioAmbients [160], Brane Calculi [20], k-calculus [47], Bio-PEPA [37], continuous π-calculus [109], β-binders [155]): all of them follow the paradigm illustrated in Table 2.1 and the have ability of handling concurrency, non-determinism, stochasticity and cooperation/competition for resources. Moreover, most of them are equipped with software tools for performing simulations – following one of the variants of the Stochastic Simulation Algorithm introduced by Gillespie [74] (see A.2 for a brief description of the algorithm) – and for analyzing their results. In Section 3 we will give a detailed description of the language that we have chosen for this work (i.e. BlenX) and its modelling approach, but here follows a concise description of other process calculi that have been developed starting from similar approaches and for similar purposes.

Stochastic π-calculus [156]: this is the first process calculus used to represent biological systems and it is derived from π-calculus [131]. It is one of the most famous representatives of the process calculi family, invented to specify and study the behaviour of concurrent software systems. Molecules are modelled as processes, and molecular complexes

are rendered by parallel compositions of processes sharing private names. Movements between complexes and formations of new complexes are represented as transmissions of private names. Once a complex is formed, its components interact by communicating on complementary sites. Two simulators for the biochemical stochastic π-calculus exist that implement the Direct Method of the Stochastic Simulation Algorithm (see Appendix A.2): BioSPI [156] and SPiM [150]. Interesting applications of stochastic π-calculus on real biological scenarios can be found in [121, 32, 108].

Bio-PEPA [37]: is an extension of the PEPA language (Performance Evaluation Process Algebra) [91] for dealing with biochemical signalling pathways. PEPA is a formal language for describing Continuous Time Markov Chain originally defined for the performance analysis of computer systems. PEPA allows one to quantitatively model and analyze large pathway systems and it is supported by a lot of software tools for analysis and stochastic simulations. A Bio-PEPA system is a formal, intermediate and compositional representation of biochemical systems, on which different kinds of analysis can be carried out: deterministic analysis of the ODE system, stochastic simulations (with the Stochastic Simulation Algorithm), generation and analysis of the underlying Continuous Time Markov Chain and generation of the input code for the PRISM model checker. The Bio-PEPA extension modifies PEPA to deal with some features of biological models, that are peculiar of those kind of systems [18]:

- Functional rates: In contrast to PEPA, individual processes are not able to define their own rates for actions. Instead the rate associated with an action is specified once, independently of the processes in which the action occurs. The value of this rate can be specified to be a function that depends on the current state of the system.
- Stoichiometry: For each action, as well as its type, the stoichiometry or degree of involvement is also specified.
- Parameterised processes: Bio-PEPA has been designed to support the population based reagent-centric style of modelling and so a model consists of a number of sequential components each representing a distinct species which evolve quantitatively (increasing or decreasing amounts). Thus in order to capture the state of a system each component is parameterised recording its current level.
- Differentiated prefix: For each action (reaction) that a component is involved in it records its role within that reaction, e.g. reactant, product, inhibitor etc. This enables the appropriate values to be used in the functional rate associated with this reaction.

Recently, some extensions of Bio-PEPA have been defined in order to represent some specific features of some biochemical networks. Specifically, the language has been ex-

tended to support SBML-events that represent changes in the system due to some trigger conditions and to support the definition of a hierarchy of compartments that have a fixed structure, but a dynamic varying size. Interesting applications of Bio-PEPA on real biological scenarios can be found at `http://biopepa.org/`.

BioAmbients [160]: it is a variant of Mobile Ambients [24] for systems biology and its focus is on biological compartments. Localization of molecules in specific compartments is extremely important in regulatory mechanisms because often a molecule can perform its task only if it is in the right compartment. Ambients can be nested and organized in a hierarchical way and (like in π-calculus) biological entities interact by means of communication, which can occur only between processes belonging to the same compartment, to parallel compartments or to compartments contained one in the other. Moreover, primitives for movement between compartments are defined. The language is equipped with a stochastic extension and a simulator, based on Gillespie's algorithm and implemented as part of the BioSPI project. Moreover in [140, 39] results concerning the applicability of Control Flow Analysis and Abstract Interpretation for the static analysis of BioAmbients models are presented.

Brane Calculi [20]: it is a calculus focused on biological membranes, which are not considered only as containers, but are active entities. A system is viewed as a set of nested membranes and a membrane as a set of actions. Brane Calculi primitives are inspired by membrane properties; membranes can merge, split, shift or act as channels. In [47], an extension called Projective Brane Calculus is presented. The goal of the extension is to refine Brane Calculi with directed actions, which tell whether an action is looking inwards or outwards the membrane. This modification brings the calculus closer to biological membranes. Recently, to improve the consistency with biological characteristics of membrane reactions, a new extension has been proposed in [50]. This extension uses a generalized formalism for action activation with a receptor-ligand type channel construction that incorporates multiple associations and a concept of affinity.

The κ-calculus [46]: is a rule-based language for modelling protein interaction networks that allows the formalization of molecular agents and their interactions in signalling networks. The κ description of a system consists of a collection of agents and rules. An agent has a name and a number of labelled sites, collectively referred to as the agent's interface. A site may have an internal state, typically used to denote its phosphorylation status or other post-translational modifications. Rules provide a concise description of how agents interact. Elementary interactions consist of the binding or unbinding of two

agents, the modification of the state of a site, and the deletion or creation of an agent. The rule-based modelling approach of κ incorporates causality constraints in the rules by using partial complexes: only the aspects of the state of a complex which matter for an event to happen need to be specified. This reliance on partial complexes allows the capture of compact descriptions and work around the huge numbers of combinations one would have to contemplate (or neglect) otherwise. It is possible to associate rate constants with each rule and the rule has to be expressed in elementary form. The language is equipped with a visual notation, where proteins are represented by boxes with domains on their boundaries. The calculus is provided with a stochastic simulator, and a series of tools (that can be found at `http://www.kappalanguage.org/`) that allow different kinds of analyses on κ-calculus models. Moreover, [112] introduces the bioκ-calculus, a calculus for describing proteins and cells which tries to unify primitives and concepts of Brane Calculi and κ-calculus.

The continuous π-calculus [109, 111]: this is a process calculus for modelling behavior and variation in molecular systems. Processes are parallel combinations of species, where species are very similar to π-calculus processes. Communication is through named channels, but there is no distinction between names and co-names. Any name can in principle communicate with any other; an *affinity matrix* specifies whether any two names can communicate and at which rate. The calculus is provided with an operational semantics in terms of real vector spaces, that offers a fully modular and compositional method of generating a set of ordinary differential equations (ODEs). The calculus is specifically designed to study evolutionary properties of biological systems.

Language for Biological Systems [148, 147]: this language is based on the Calculus of Biochemical Systems (CBS) and combines rule-based approaches to modelling with modularity. It allows the description of metabolic, signalling and regulatory networks in terms of reactions between complexes in a concurrent way and inside a hierarchy of compartments and with possible cross-compartment interactions and transport. Additional features of LBS include expressions for manipulating large complexes in a concise manner, parameterised modules with a notion of subtyping for writing reusable modules, and nondeterminism for handling combinatorial explosion. Formal specification of LBS are given both through an abstract syntax and a specific semantics such as Petri nets, coloured Petri nets, ODEs and continuous time Markov chains.

β-binders [155]: in this formalism processes are encapsulated into boxes with typed interfaces. Types represent the interaction capabilities of the boxes. β-binders aims to

enable non-determinism of communication by introducing the concept of *compatibility* [152], a notion that extends the *key-lock* notion of complementarity between actions and co-actions typical of process calculi where the precise matching between an input and an output over a given channel is always required. In β-binders, boxes are able to perform complementary input/output actions over one of their interfaces only if the types of the involved interfaces have some rate of compatibility. In this way, whichever notion of type compatibility is assumed, the communication ability of boxes is mainly determined by the types of their interfaces rather than by the actual naming of the relevant input and output actions. The formalism is also provided with *join* and *split* operations, i.e. parametric rules that drive the merging and splitting of boxes depending on their structure. A stochastic extension of β-binders for quantitative experiments is presented in [52].

The list of process calculi presented here is by no means exaustive but we chose to cite the most representative languages applied on real biological case studies and the ones closer to the approach used by the language chosen for developing our study and explained in the following chapters.

2.2 Inference of Biological Systems

The data currently collected in high-throughput experiments on genomic, proteomic, and metabolomic scales hold the promise of identifying not only the components but also the interactions which comprise large-scale biochemical networks. Parameter estimation (also known as model calibration) from experimental data is a bottleneck for a major breakthrough in computational systems biology in the present post-genomic era. Given a model structure and a set of experimental data, the objective of parameter estimation is to calibrate the model (looking for parameters which are rarely measured directly) so as to reproduce the experimental results in the best possible way. This calibration is performed by minimizing a cost function which measures the goodness of the fit. In other words, the identification problem is stated as the optimization of a scalar cost function with respect to the model parameters. The cost function is usually a certain weighted distance measure between the experimental values correponding to the measured variables and the predicted values for those variables. The optimal value will of course depend on the cost function chosen. Cost functions that have been shown to work well in practice include Bayesian, maximum likelihood and least squares estimator. The selection criterion will depend on the assumptions about the data disturbance and on the amount of information provided by the user. In most approaches dealing with parameter estimation the cost function is the likelihood function, also know as joint transitional density, that expresses the probability of obtaining the observed outcomes in terms of measured systems vari-

ables and parameters. Thus it can be used to determine unknown parameters based on known outcomes. The optimal values of the parameters can be estimated by maximizing the likelihood function (maximum likelihood criterion). There are two general methods of parameter estimation: the least-squares estimation (LSE) and the maximum likelihood estimation (MLE).

Least Squares Estimator [15]: The method of least squares is a standard approach to the approximate solution of overdetermined systems. It is a popular technique of model fitting and is tied to many familiar statistical concepts such as linear regression, sum of squares errors, and root mean squared deviation. Least squares problems fall into two categories: linear least squares and nonlinear least squares, depending on whether or not the residuals (i.e. the distances between data in the starting dataset and the predicted value) are linear in all unknowns. In both cases the aim is to find the parameters which minimize the sum of squared distances between the observed responses in the dataset, and the responses predicted by the approximation. Least squares corresponds to the maximum likelihood criterion if the experimental errors have a normal distribution and can also be derived as a method of moments estimator.

Maximum likelihood [149]: Maximum likelihood estimation, instead, is a popular statistical method that starts from a generative assumption on the probability distribution underlying the data, and estimates parameters via an optimisation procedure.

A maximum likelihood estimator coincides with the most probable Bayesian estimator given a uniform prior distribution on the parameters. MLE has two optimal properties: *consistency*, i.e. the ability, for data of sufficiently large samples, to asymptotically recover true parameter values that generated the data; and the *parameter invariance*, i.e. same MLE solutions can be obtained independently of the parametrization used. Moreover, MLE is useful for inference with missing data and modelling random effects, as well as having close ties to more general Bayesian methods. The main idea behind the method is the following: suppose there is a sample $x_1, x_2, \ldots, x_n$ of n independent and identically distributed observations, coming from parametric model $f_0 = f(\cdot|\theta_0)$. Given sample of data with observed values $x_1, x_2, \ldots, x_n$, the likelihood function of the parameter can be expressed as: $\mathcal{L}(\theta \,|\, x_1, \ldots, x_n) = f(x_1, x_2, \ldots, x_n \mid \theta) = \prod_{i=1}^{n} f(x_i|\theta)$. In practice it is often more convenient to work with the logarithm of the likelihood function ($\hat{\ell}$) called the log-likelihood. The method of maximum likelihood estimates θ_0 by finding a value of θ that maximizes $\hat{\ell}(\theta|x)$: $\hat{\theta}_{\text{mle}} = \underset{\theta \in \Theta}{\arg\max}\, \hat{\ell}(\theta \,|\, x_1, \ldots, x_n)$.

Bayesian inference [81, 195]: Bayesian inference is another approach to statistical inference. Bayesian inference is based on the Bayes' theorem which states that:

$$P(\Theta|D) = \frac{P(\Theta) \cdot P(D|\Theta)}{P(D)}$$

where Θ represents the set of parameters of the model, $P(\Theta)$ is called the *prior probability* of Θ that is deduced before data, D, became available, $P(D|\Theta)$ is the conditional probability of seeing the data D with the choice of parameters Θ, $P(D)$ is a normalizing factor that represent the model likelihood (i.e. the a priori probability of witnessing the evidence D under all possible choices of parameters) and finally $P(\Theta|D)$ is the *posterior probability* of Θ given D. Bayesian statisticians believe that even with different prior probabilities, new evidence from repeated observations will tend to bring their posterior subjective probabilities closer together. However, others argue that with widely different prior probabilities the posterior probabilities may never converge even with repeated collection of evidence and that priors which are completely different initially can remain completely different over time despite accumulation of evidence [100]. Parameter estimation using Bayesian strategies correspond to finding the parameter of the model that minimizes the posterior expected value of a loss function (i.e. the posterior expected loss) or maximizes the posterior expectation of a utility function.

The underlying inference principles of MLE and Bayesian inference, even if related, are different. Let us consider a model described by some parameters $\vec{\theta} = \{\theta_1, \ldots, \theta_N\}$, that are free to vary. In a Bayesian approach, we have first to formulate some prior beliefs about the value of these parameters, These beliefs are formalised by a probability distribution on the parameters, $P(\vec{\theta})$. Once the prior beliefs have been formulated, the likelihood of the observed data is used to update the prior probability distribution to a posterior probability distribution. The posterior probability distribution essentially represents the end result of a Bayesian analysis: unlike maximum likelihood analysis, the aim of a Bayesian analysis is not to provide so-called *point estimates* of the model parameters, the result of the analysis is the posterior probability itself. If we are interested in one parameter, we would calculate the marginal probability distribution for that parameter by integrating (or summing) out all other parameters in the model ($P(\Theta_i|D) = \int_{-i} P(\Theta|D) d\Theta_{-i}$). The marginal distribution for a parameter is identical up to a normalizing constant to the integrated likelihood that would result if we were to integrate out all parameters in the likelihood function except the one we focused on. Consequently, the maximum posterior probability estimate is the same as the maximum likelihood estimate when a pure integrated likelihood approach is used. Other differences between maximum likelihood inference and Bayesian inference happen when analysing multi-parameter models: the

maximum likelihood approach distinguishes between structural and nuisance parameters while there is no such difference in Bayesian analysis. The result of a Bayesian analysis is a joint probability distribution on all parameters in the model [84, 165].

When estimating parameters of dynamical systems with optimization methods a number of difficulties and pitfalls may arise, e.g. convergence to local solutions (if standard local methods are used and especially when only bad starting values for parameters are available), very flat objective function in the neighborhood of the solution, overdetermined models (leading to many solution vectors), badly scaled model functions and non-differentiable terms in the systems dynamics. The recent literature reports many examples of effective methods attempting to work out the aforementioned difficulties of parameter estimation. Here we briefly mention the most recent ones that apply both to determinsitc and stochastic biochemical processes.

Polisetty et al. in [151] suggested global optimization techniques as an alternative to traditional local methods. Rodrigez-Fernandez et al. in [164] developed a hybrid stochastic-deterministic global optimization method. Moles et al. in [133] explored several state-of-the-art deterministic and stochastic global optimization techniques and compared their accuracy and effectiveness on nonlinear biochemical dynamic models. Tian et al. [182] presented a simulated maximum likelihood method to evaluate parameters in stochastic models described by stochastic differential equations. They propose different types of transitional probability and a genetic optimization algorithm to search for optimal reaction rates. Chou et al. [33] developed an alternate regression method, that dissects the parameter inference problem into iterative steps of linear regression. Sugimoto et al. [179] provided a computational technique based on genetic programming that simultaneously generates biochemical equations and their parameters from time series data. Reinker et al. [163] are the authors of the approximate maximum likelihood method and the singular value decomposition likelihood method that estimates stochastic reaction constants from molecule count data measured with errors at discrete time points. Finally, we mention the works of Quach et al. and Vyshemirsky and Girolami [158, 193], that developed Bayesian model-based inference techniques.

The estimation of parameters can also be done taking advantage of the numerous learning approaches that has been recently applied to parameter inference in dynamical systems. Methods like Simulated Annealing [190] have been used in parameter estimation for biological network dynamics in [162, 34], for fitting parameters in gene regulation networks, different kinds of evolutionary programming where used in similar contexts (reviews comparing the different approaches to standard chemical kinetic models can be found in [132, 130]). Approximate Bayesian computation (ABC) algorithms are another

group of methods that recently have gained a lot of attention for performing Bayesian inference without the need for explicit evaluation of the model likelihood function [11, 127, 174]. The algorithms can be used with computer models that generate sample data sets rather than return likelihoods, so they enable inference using only simulations generated from the model: the main idea is to replace the calculation of the likelihood with a comparison between the observed and simulated data [196]. Synthetic experimentation using stochastic simulations has been shown to be useful also to predict the effects of parameter changes in stochastic models. The authors of [69] presented a framework that, thanks to the usage of a parallel architecture, generates and explores the parameter space of a model allowing parameter inference and multivariate analysis in an efficient way. Moving beyond the usage of SSA to sample the joint probability distribution derived by the chemical master equation describing the system, [194] proposed a general formula for the gradient of the likelihood function of stochastic models of biochemical networks given a discrete time-course using reversible jump Markov chain Monte Carlo sampling: the gradient has then been used to estimate the parameter values with a gradient descent method.
All these kinds of approach, however, are computationally intense because they iterativelly modify the model's parameters: according to specific rules that depend on the chosen method, they decide if the choice of parameter is correct (using some kind of acceptance criteria) and, if this is not the case, they generate another set of parameters to be tried (using selection strategies that again depend on the specific method chosen). In the majority of the application papers of these methods convergence issues are raised and the choice of the acceptance criteria and/or the iterative step need to be carefully tuned by the user. In spite of these possible disadvantages, these approaches are gaining popularity in the context of inference of biological systems so they are a promising research area in which more work is required to understand how the different options affect the final results in order to help and guide the user in the choice of which method is more suitable for which specific kind of data/model.

Tools for parameter estimation can be found as an integral part of some simulation tools (e.g. Matlab packages, Copasi [95]), but there exist also stand-alone software exclusively designed for that purpose, like PET (Parameter Estimation Toolkit, http://mpf.biol.vt.edu/pet/). The disadvantages of the most of the current tools for parameter estimation is the lack of robustness to the noise and the absence of any estimate of experimental error in their outcome. Experimental uncertainties on parameters propagate from the measurements of the concentrations of the species. Returning the parameters with an estimate of their uncertainty is essential if we want to use the tool in the context of optimal experimental design. Moreover, the majority of the current tools based

on optimization techniques, ask the user to provide some kind of *a priori* knowledge to the optimization algorithm in order to limit the region of parameter space in which to perform the search for the maximum. Further discussions about the main existing tools for parameter estimation can be found in Section 6.5.

2.3 Cell cycle models (*Saccharomyces cerevisiae*)

The general cell theory was developed in the 1830s by Theodor Schwann and Matthias Schleiden [170, 171]. It states that all living organisms are composed of nucleated cells that are the functional units of life and that cells arise only from pre-existing cells by a process of division. According to current knowledge about the structure of a cell, we know that a cell is surrounded by a membrane that separates it from its external environment and fundamental to eukaryotic cells – in contrast to prokaryotic cells – is their subdivision by intracellular membranes into distinct compartments (see Figure 2.1).

Because of the fact that growth and reproduction are major characteristics of life, crucial for these is the cell division process by which one cell divides into two and all parts of the mother cell are distributed to the daughter cells. Even if the concept of cells as the "functional units of life" was established by the mid-nineteenth century and despite the fact that from that time an enormous amount of biological data has been produced, we are still far from a detailed understanding about how cells function. Even if the physiological behavior of cells can be observed and measured by various experimental techniques, DNA can be sequenced and some molecular interactions might be detected, yet our knowledge about the mechanisms describing the observed properties is far from complete.

Mathematical modelling of the cell cycle has been started already when scientists were only guessing the mechanism that could drive the processes of cell division. From the 1960's we can find mathematical models that explain some key aspects of cell cycle regulation from phenomenological observations on cell size and cell cycle time distributions [107]. At the same time, chemical oscillators have been discovered: the mixture of some reactants discovered by Belousov and later rediscovered by Zhabotinsky served as the first and classical example of non-equilibrium thermodynamics [198]. The details of the Belousov-Zhabotinsky reaction (BZ reaction) provoked widespread interest and research in the 1970s and gave huge contributions to research on theoretical physical chemistry and later to mathematical biology. Researchers, like Albert Goldbeter, John J. Tyson, Arthur Winfree and others realized that the discoveries on chemical oscillations and the tools that have been developed for their descriptions might help to understand biological oscillations. They investigated biological systems from calcium oscillations, circadian

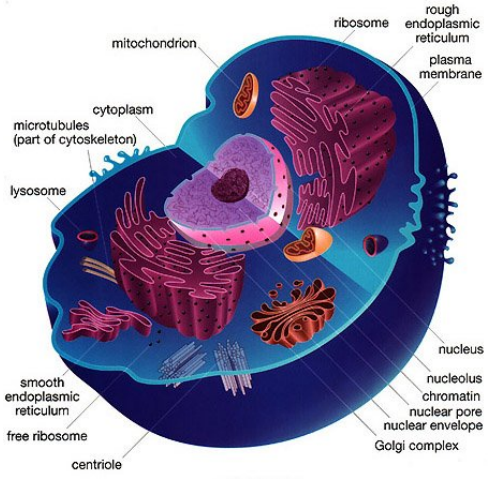


Figure 2.1: Cells are surrounded by a plasma membrane, which is made of a double layer of phospholipids. Within this membrane the cytoplasm encloses the organelles of the cell. The nucleus of eukaryotic cells contains the genetic material which chemically directs all of the cell's activities. Usually this is in the form of long strands of chromatin made of DNA and affiliated proteins. When a cell is ready to divide, the chromatin coils and condenses into individual, distinguishable chromosomes. Another organelle of the cell is the endoplasmic reticulum (ER for short). There are two kinds of ER: smooth ER and rough ER. Typically ER closer to the nucleus is rough and that farther away is smooth. Smooth ER is a transition area where chemicals like proteins the cell has manufactured are stored for transportation elsewhere in the cell. Pieces of the smooth ER called vesicles pinch off from the smooth ER and travel other places in the cell to transfer their contents. Rough ER gets its name because it has other organelles called ribosomes attached, which give it a rough appearance when viewed by an electron microscope. Rough ER and its associated ribosomes are involved in protein synthesis, with the new polypeptide being threaded into the lumen of the ER as it is formed. Ribosomes are special organelles that are directly involved in protein synthesis. They are made of RNA (ribonucleic acid) and protein and are manufactured in the nucleus (from a DNA template), then go out into the cytoplasm to function. Vacuoles and vesicles are storage organelles. Generally, vacuoles are larger than vesicles and the latter are small enough and mobile enough that they are often used to move chemicals to other locations in the cell where they might be needed. One of the places to which vesicles travel is the Golgi apparatus that is like the shipping and receiving department of the cell. Materials are received as vesicles unite with the Golgi apparatus, and sent elsewhere as other vesicles pinch off. Materials are temporarily stored in the Golgi bodies, and some further chemical reactions do take place there. Other organelles found in nearly all eukaryotic cells are mitochondria: they burn sugar for fuel in the process of cellular respiration and they are the "engine" of the cell. Mitochondria consist of a smooth outer membrane and a convoluted inner membrane separated by an intermembrane space. As sugar is burned for fuel, a mitochondrion shunts various chemicals back and forth across the inner membrane. The organelle that is responsible for the shape and the structure of the cell is the cytoskeleton that is made of various types of special proteins. Microtubules are hollow tubes made of globular proteins. Animal cells typically have a pair of centrioles located just outside the nucleus and oriented at right angles to each other. These function in cell division. Microfilaments are also part of the cytoskeleton and are made of solid rods of globular proteins. (Picture taken from http://www.animalport.com/animal-cells.html)

clocks and many others, among them the oscillations that drive cell division. As some data on the key regulator of cell cycle (CDK) that works in a complex with a cyclically appearing molecule (cyclin) were found by Nurse [143], Hartwell [86] and Hunt [63] in

the 70s-80s (breakthrough results for which they received the Nobel Prize in 2001), theoreticians started to create models to understand how CDK can regulate cell cycle events [79, 181, 186]. Some of these works laid down crucial details and basic properties of cell cycle regulation. Further experiments on yeasts and frog eggs produced a molecular description of the protein interaction network of these systems, inspiring further mathematical analysis. This success story created a great interest for modelers to ask questions in cell biology and to answer them with the help of an inter-disciplinary work of mathematicians, physicists, and other scientists [43].

Budding yeast is perhaps the most useful yeast owing to its use since ancient times in baking and brewing. The molecular machinery of eukaryotic cell cycle control is known in more detail for this yeast species (whose scientific name is *Saccharomyces cerevisiae*, from Latinized Greek *saccharo-* (sugar) *myces-* (fungus) and from Latin *cerevisiae* (of beer)) than for any other organism. Moreover, informational pathways in yeast are remarkably similar to those in fly, worm and humans, and many orthologous genes can be identified across these species.

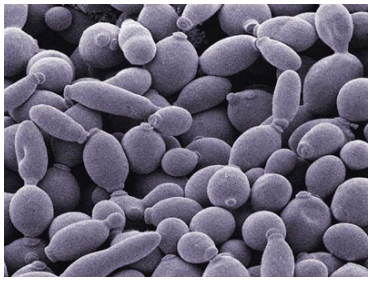


Figure 2.2: Budding yeast is the microorganism behind the most common type of fermentation. Its cells are round to ovoid, 5-10 micrometres in diameter. Its genome is composed of about $12.5 * 10^6$ base pairs and 6275 genes, compactly organized in 16 chromosomes. It reproduces by a division process known as budding. (Picture taken from http://rushartsbiology.wikispaces.com/Visuals+-+Unit+5)

The reason why this organsim became the eukaryotic model organism is because it scores favorably on the following criteria:

- as a single celled organism, it is small with a short generation time (doubling time 1.5-2 hours at 30°C) and can be easily cultured. These characteristics allow for the quick production and maintenance of multiple specimen lines at low cost;
- it can be easily manipulated genetically, allowing for either the addition or deletion of new genes;
- as a eukaryote, it shares the complex internal cell structure of plants and animals without the high percentage of non-coding DNA that can confound research in higher eukaryotes.

For many years, molecular biologists have meticulously characterized individual components of budding yeast cell cycle to derive a consensus picture of the regulatory network. One of the most common ways to describe this network of biochemical reactions is to translate the interactions of the hypothetical molecular wiring diagram into ordinary differential equations. The solutions of ODEs can be compared with existing experimental data and can be used both to verify a hypothesized mechanism that controls the system and to provide testable predictions. ODEs have been used for many years to create mathematical models of eukaryotic cell cycle regulation since this was the technique that was used to investigate chemical and physical oscillators, thus all the developed tools of dynamical systems theory could be used to analyze ODE models of cell cycle regulation.

Focusing our attention on the history of budding yeast cell cycle modelling, we can identify one of the first version of the model in [31] where the authors hypothesize the existence of an hysteretic switch that controls the entry into S phase (for explaination of the biological meaning of the different phases/roles of proteins in the cell cycle process, see Section 4.1.1): this prediction and others were tested experimentally by Fred Cross's group in a seminal paper that might be the first case when a molecular genetics lab focused solely on verifying a mathematical model of cell-cycle regulation [42]. Later the groups joined forces to create a model that can simulate the behavior of more than 120 mutants of the budding yeast [30]. This model also predicted the existence and regulation of a phosphatase that later was identified [159]. Recently other groups have presented their own models of the budding yeast cell cycle, focusing on various aspects of the regulatory system: the authors of [14] use a genetic network model of 11 genes that coordinate the cell cycle process to study a dynamical attractor that results in a remarkable robustness against noise fluctuations; in [7] the authors report a mathematical model of the G1 to S network that newly takes into account nucleo/cytoplasmic localization, the role of the cyclin-dependent kinase Sic1 in facilitating nuclear import of its cognate Cdk1-Clb5, Whi5 control, and carbon source regulation of Sic1 and Sic1-containing complexes; [176] analyze the dynamical properties of the intertwined feedback loops that drive the cell cycle alternation of states; [177] discuss aspects of model development using the example of cell cycle regulation in yeast in order to suggest that capturing complex dynamic networks is feasible despite incomplete quantitative biological knowledge.

Most of the models mentioned above use the mathematical formalism of ODEs to describe the dynamics of the system, and there exist many mathematical analysis tools able to track the steady states and dynamical transitions of cell-cycle control system [9, 44]. As the complexity of the known cell-cycle regulatory network increased in the last few years, logical dynamic modelling [180] and especially Boolean algebra became another fashionable modelling approach, because it is a very simple formalism where the activity

of each component is represented by two states: ON and OFF. The success of Boolean algebra for studying budding yeast cell cycle might be partially influenced by the results presented in [122], where the authors showed in a logical model that trajectories from 86% of all possible initial states lead the system into one state representing G1-phase of the cell cycle. Most of these trajectories funneled into a path which steps through the different phases of the cell cycle, showing that the cell cycle is robustly designed.

Although some of these logical models were already introducing a stochastic treatment of the system, recently some more expressive and specific formalisms have started to consider the effects of molecular noise in the cell cycle regulatory network. These stochastic models can investigate how individual cells might differ from the average behavior of the population (the output of deterministic ODE models). Stochastic fluctuations could be relevant for certain mutant cell populations that show partial viability [135]. Furthermore, recent advances in experimental observations on single cells allow the measurement of the distribution of behaviors in a population of cells, for example, the measurements of the noisiness of the G1/S transition in budding yeast cells provided by Cross's group [10, 57]. Recently [101] explored the role of noise by measuring the cell size and cell cycle time distributions in a budding yeast cell cycle model. In order to investigate the stochastic property of the reaction network, the authors unpacked the ODE model in [187], with all its complicated functions, into elementary reaction steps. The stochastic models described so far use the mathematical formalism of stochastic differential equations but recently some modelling concepts expanded from computer science towards biological systems proved to be useful in modelling budding yeast cell cycle. [135] used the formalism of Petri nets (which are a mathematical modelling language defined as directed bipartite graphs, in which the nodes represent transitions (i.e. events that may occur) and places (i.e. conditions); the directed arcs describe which places are pre- and/or postconditions for which transitions) to show that the stochastic fluctuations could be relevant for certain mutant phenotypes that show partial viability: these kinds of analysis could not be carried on with deterministic models at population level. Finally, as shown in the previous section, languages derived from computer science have been adapted to model biological systems and different rule-based modelling languages has been used to build models of the cell cycle: [116] chose to implement, using stochastich π-calculus, the control mechanism of the cell cycle (modelled by Novak in 1999) that is accounting for the antagonistic interaction between cyclin-dependent kinases dimers and the anaphase promoting complex; [37] built in Bio-PEPA the model, presented by Goldbeter in 1991 and later extended by Gardner et al. in 1998, that describes the negative feedback loop obtained by the fact that cyclins promotes the activation of a cdk (cdc2) which in turn activates a cyclin protease which in turn promotes cyclin degradation.

Part I

The modelling framework

Chapter 3

Modelling with BlenX

3.1 The language

BlenX is a stochastic programming language explicitly designed to model interactions of biological entities. Stochastic means that quantitative information about speed and probability of actions is provided with systems specifications. A BlenX program is made up of a program file for the system structure (a textual file, usually with extension .prog), an interfaces file for the quantitative information about the system (a textual file, usually with extension .types) and an optional declaration file for the user-defined variables and functions (a textual file, usually with extension .func). A BlenX program can be given as input to the Beta Workbench [55, 56] (BetaWB for short) for running stochastic simulations of the model. BetaWB implements an efficient variant of the Gillespie algorithm [74] allowing the user to have stochastic time courses of the system under consideration.

For a detailed description of the BlenX languange and BetaWB functionalities we refer the reader to [55, 53]. Here we just sketch the main features of the language used for the translation procedure presented in section 3.2 and the main peculiarities of the framework that allow us to perform interesting analysis of the chosen real biological case study (i.e. the budding yeast cell cycle).

The underlying metaphor of BlenX is that a *biological entity*, i.e. a component that is able to interact with other components to accomplish biological functions, is represented by a computational device, a *box*, composed of a set of interfaces (also called binders) and an internal program. Interfaces have associated types (representing their state) and are the places where an entity interacts with other entities; the internal program, instead, codifies for the mechanism that transforms an interaction into a conformational change of the entity, e.g. the modifications of box's interface types. Figure 3.1 shows the graphical notation we use to represent boxes. The program file contains the definition of all the

boxes acting in the model.

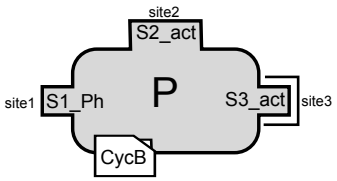


Figure 3.1: The small squares on the border of the box are the binders; site1, site2 and site3 are the interfaces subjects (omitted when not necessary); S1_Ph, S2_act and S3_act are the interfaces types; the line surrounding the interface with type S3_act indicates that the interface is hidden; P is the internal program and CycB is the name of the box.

A box is defined using the keyword *bproc* and, for instance, the following piece of code defines the box in Fig. 3.1:

```
let CycB : bproc = #(site1,S1_Ph), #(site2,S2_act), #h(site3,S3_act)[ P ];
```

#(site1,S1_Ph), #(site2,S2_act), #h(site3,S3_act) are box's interfaces defined by a subject (e.g. site1) and a type (e.g. S2_act). Subjects and types must all be different. Any interface has an associated state that can be free (denoted with #), hidden (denoted with #h) or bound. Subjects can be referred by the internal program P, that is described by a process built on top of the following set of primitives, where the first two are derived from the classical process algebras primitives constructs while the others are specific for the BlenX language:

- Input: `x?(y) / x?()`
- Output: `x!(y) / x!()`
- Change: `ch(x,A) / ch(r,x,A)`
- Delay: `delay(r)`
- Die: `die / die(r)`
- Expose: `expose(x:s,A) / expose (r,x:s,A)`
- Hide: `hide(x) / hide(r,x)`
- Unhide: `unhide(x) / unhide(r,x)`

The input action intuitively means that a process is willing to receive on the channel `x` a name that will replace the target variable y. The output action sends a name `y` along channel `x`. When both the name sent and the target variable are irrelevant (i.e. we want to represent just pure synchronization and no information exchange), we omit the name `y`. The change action changes the type of an interface with subject `x` into `A`. The *delay* action represents a time delay, while the *die* action eliminates the box

executing it. The directive *expose* allows the exposure of a new interface, while *hide* and *unhide* actions modify the state of an interface to hidden or free, respectively. Notice that for most of the actions listed above it is also possible to define a specific rate `r` associated to the action. The basic actions can be combined by sequential composition (“.” symbol), parallel composition (“|” symbol), choice (“+” symbol), replication (*rep* keyword) and conditional “if/then” statement. These operators allow us to construct complex programs describing the internal behavior of a box. The choice operator “+” composes two processes that can be alternatively executed, i.e. the execution of one of the processes prevents the execution of the others. The parallel operator “|” composes two processes that execute concurrently and allows them to communicate or synchronize when they perform complementary input/output actions on the same channel. The *rep* operator produces as many copies of its argument in parallel as needed (i.e. possibly infinitely many).

In addition to the classical process algebra actions described above, BlenX allows the definition of events which are statements, or verbs, that are executed with a specified rate and/or when some conditions of the system are satisfied. A single event is the composition of a condition *cond* and an action *verb*: `when (cond) verb`. Conditions are used to trigger the execution of an event when some elements are present in the system, when a particular condition is met, with a given rate, or at a precise simulation time or simulation step. Events can split an entity into two entities, join two entities into a single one and add/remove entities into/from the system. For an intuitive explaination of the main primitives of the language described up to now, see Figure 3.2.

Another characteristic of BlenX concerns the possibility of separating the qualitative description of the structure of the model (encoded in the program file) from the quantitative information about the rates of the different reactions (encoded in the declaration file). In the latter file the user defines:

- constants: identifiers associated to real values that cannot be changed at runtime;
- variables: identifiers that can assume modifiable real values. The content of a variable is automatically updated when the evaluation of the expression that it is defining it changes; the content of the variable can also be changed by an *update* event;
- continuous variables: identifiers that depend on time and whose value is determined by an expression which is recalculated at fixed time steps;
- functions: complex mathematical expressions that are usually used as rates for non-elementary reactions (coded as events, for modelling an aggregated process or when the precise elementary steps of interaction between entities are not known).

The last peculiar characteristic of the BlenX language are *templates*. They are a feature of many programming languages that allow us to code, in an extended grammar,

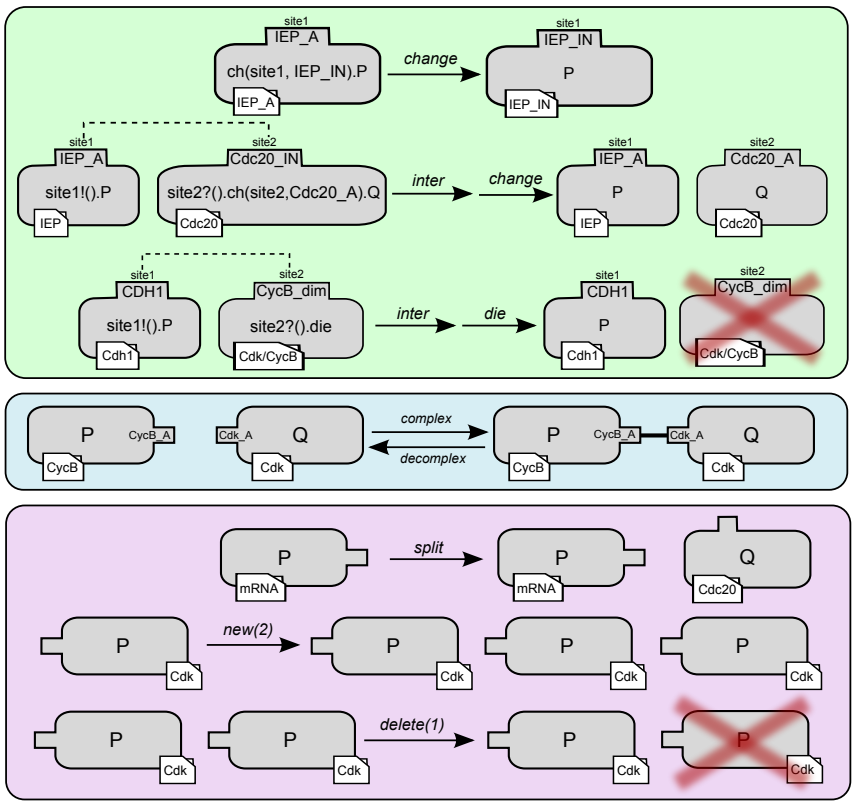
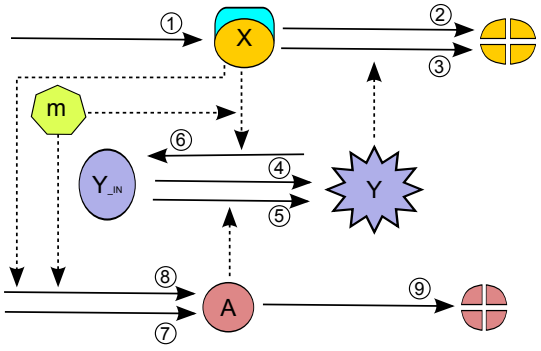


Figure 3.2: **Intuitive behavior of some BlenX primitives.** Each row represents a different primitive. The first three primitives are internal actions, the fourth is a complexation/decomplexation event and the last three are global events on boxes. The first action is a *change* action that allows the modification of the type of a binder from the internal code of the box. The second row shows a sequence of two actions: an intercommunication between boxes with compatible interfaces and the change of an interface performed by the box that received the communication message (modelling the effect of IEP on the activation of Cdc20 box). Analogously the third row shows a sequence of an intercommunication action followed by a *die* action performed by one of the two boxes (modelling the fact that Cdh1 is able to degrade Cdk/CycB species). The *complex* event is performed when two boxes have interfaces with compatible types and the user wants to model the creation of a private and permanent (until the complementary decomplexation event) communication channel between two specific boxes. Those two boxes will maintain their individual internal behavior but they will be able to communicate on the shared channel. Finally the last three rows are the graphical representation of the events of *split*, *new* and *delete*. The *split* event is used to substitute one box with two (possibly different) boxes, while the last two events model the creation/deletion of an arbitrary number of boxes in the system.

parametric processes, which can contain variable parts instantiated later by the compiler with respect to the base grammar. In BlenX, template code is specialized and instantiated at compile time using interfaces identifiers, code or names that are passed as template arguments. In general, in a big and complex model, the usage of templates reduces the amount of code that needs to be written making the whole model more modular and easy

to modify. A way to reduce the code that has to be written is identifying "structurally equal" mechanisms in the model: for example, in cell cycle models there are a lot of species whose degradation is the result of the interaction of another species with a mass action kinetic law, or different proteins have a self-degradation mechanism that only differs in the rate constant that is driving the process. Thus, in BlenX any common pattern of this kind can be grouped and coded in an easy way through the templates.

In order to show how to use the BlenX features explained above concretely, we present a "toy model" of the budding yeast cell cycle that will be used in Section 4.2 for performing quantitative analysis of the irreversibility of the cell cycle process (for details about the biological meaning of this model, we refer to Chapter 4).



$$
\begin{aligned}
\frac{d|X|}{dt} &= \underbrace{k1/\alpha}_{(1)} - \underbrace{k2p \cdot |X|}_{(2)} - \underbrace{k2s \cdot \alpha \cdot |X| \cdot |Y|}_{(3)} \\
\frac{d|Y|}{dt} &= \underbrace{\frac{k3p \cdot \alpha |Y\_IN|}{J3 + \alpha \cdot |Y\_IN|}}_{(4)} + \underbrace{\frac{k3s \cdot \alpha \cdot |Y\_IN| \cdot |A|}{J3 + \alpha \cdot |Y\_IN|}}_{(5)} + \\
&\quad \underbrace{- \frac{k4 \cdot m \cdot \alpha \cdot |Y| \cdot |X|}{J4 + \alpha |Y|}}_{(6)} \\
\frac{d|Y\_IN|}{dt} &= \underbrace{- \frac{k3p \cdot \alpha |Y\_IN|}{J3 + \alpha \cdot |Y\_IN|}}_{(4)} \underbrace{- \frac{k3s \cdot \alpha \cdot |Y\_IN| \cdot |A|}{J3 + \alpha \cdot |Y\_IN|}}_{(5)} + \\
&\quad \underbrace{+ \frac{k4 \cdot m \cdot \alpha \cdot |Y| \cdot |X|}{J4 + \alpha |Y|}}_{(6)} \\
\frac{d|A|}{dt} &= \underbrace{k5p/\alpha}_{(7)} + \underbrace{- \frac{k5s \cdot (m \cdot |X|)^n \cdot \alpha^{n-1}}{J5^n + (m \cdot \alpha \cdot |X|)^n}}_{(8)} \underbrace{- k6 \cdot |A|}_{(9)} \\
\frac{dm}{dt} &= \mu \cdot m \cdot (1 - m/mstar)
\end{aligned}
$$

Figure 3.3: The antagonism between Cdk/CycB complex (X) and the Cdh1/APC (Y) that drives cell cycle oscillations in budding yeast as explained in [142] (graphical representation on the left and ODE system on the right). For an explanation of the biological meaning of the different entities, see Chapter 4. Solid arrows link reactants to products, dashed lines represent the mediation effect that some species have on reactions. Reactions are numbered to allow an easy reference from the code in Table 3.1.

Table 3.1 shows the input file for the model depicted in Figure 3.3. The reactions in the table are numbered as in the figure. In general, synthesis mechanisms are modelled with a *new* event, while degradation ones are modelled with *delete* events. All the events have an associated rate function (usually with a complex mathematical expression) that is defined in the .func file and then its name used in the appropriate event. In the .func file, two variables are defined: `m` (which represents the mass of the cell), that is modelled as a

```
//.PROG
[steps = 2000, delta = 0.2]
let X : bproc = #(x,X)[ nil ];
let Y : bproc = #(y,Y)[ nil ];
let Y_IN : bproc = #(y_in,Y_IN) [ nil ];
let A : bproc = #(a,A)[ nil ];

when(X :: X_synthesis) new(1);
when(X :: X_self_degradation ) delete(1);
when(X :: X_degraded_by_Y ) delete(1);
when( Y_IN :: Y_self_activation ) split(Nil, Y);
when( Y_IN :: Y_activation_with_A ) split(Nil, Y);
when( Y :: Y_deactivation ) split(Nil, Y_IN);
when(A :: A_synthesis ) new(1);
when(A :: A_collaborate_M_X ) new(1);
when(A :: A_self_degradation ) delete(1);

when (:mCycB->0.2,mCycB<-0.1:) update (m,mass_div);

run 4 X || 424 Y || 424 A
```

```
-> Simulation output length-accuracy
->--
| Definition of the
| main species (boxes)
->--

-> Reaction (1)
-> Reaction (2)
-> Reaction (3)
-> Reaction (4)
-> Reaction (5)
-> Reaction (6)
-> Reaction (7)
-> Reaction (8)
-> Reaction (9)

-> m division controlled by the state of mCycB

-> Initial conditions for the simulation
```

```
//.FUNC
let mu :  const = 0.005; let k1 : const = 0.04; let k6 : const = 0.1;
let k2p : const  = 0.04; let k2s : const = 1; let mstar : const = 10;
let J3 : const = 0.04; let k3p : const = 1; let J5 : const = 0.3;
let k3s : const = 10; let k4 : const = 35; let k5s : const = 0.2;
let J4 : const = 0.04; let k5p : const = 0.005; let n :  const = 4 ;
let alpha : const = 0.00236012;

let m(0.1): var = mu * m * (1 - m/mstar) init 0.45;

let mCycB : var = m * |X| * alpha;

let mass_div : function = m / 2;

let X_synthesis: function = k1 / alpha ;
let X_self_degradation : function = k2p * |X|;
let X_degraded_by_Y : function = k2s*alpha * |X|*|Y|;
let Y_self_activation : function = (k3p * |Y_IN|)
                           / (J3 + alpha * |Y_IN|);
let Y_activation_with_A:function=(k3s*alpha*|A|*|Y_IN|)
                                /(J3 + alpha * |Y_IN|);
let Y_deactivation : function =(k4 * m * alpha*|X|*|Y|)
                                   /(J4 + alpha * |Y|);
let A_synthesis : function = k5p / alpha;
let A_collaborate_M_X:function = (k5s / alpha)
                     / (pow((J5/(m*alpha*|X|)),4)+ 1);
let A_self_degradation : function = k6 * |A|;
```

```
->--
|
| Definition of the constants
|
->--

-> Def. of the continuous variable of the mass

-> Def. of discrete variable that triggers m division

-> Def. of the expression that divides m

-> Reaction (1) kinetic rate function
-> Reaction (2) kinetic rate function
-> Reaction (3) kinetic rate function
| Reaction (4) kinetic rate function
-^--
| Reaction (5) kinetic rate function
-^--
| Reaction (6) kinetic rate function
-^--
-> Reaction (7) kinetic rate function
| Reaction (8) kinetic rate function
-^--
-> Reaction (9) kinetic rate function
```

Table 3.1: BlenX input files of the budding yeast cell cycle toy model in Fig. 3.3. The left column shows the content of the textual .prog and .func file, and the right column contains a small description of the different lines of code. See text for details.

continuous variable (recalculated every 0.1 minutes (simulation time) with the expression provided and initialised to 0.45) and a discrete variable which represents the combined effect of X (CycB) and of the mass (`mCycB`). This variable is automatically updated when the evaluation of its defining expression changes (i.e. when the amount of X boxes changes during the simulation). `mCycB` variable is used to trigger the cell division (i.e. the *update*

event in the .prog file): in particular this event will be immediately fired whenever the sequence of conditions that guard the event are met. Those conditions are based on the traversal of successive states which are examined in sequence. The *update* event is fired when all the states, in sequence, are met throughout the system dynamics. In our specific example we want to monitor the crossing of "`mCycB`" value 0.2 from below and then 0.1 from above (i.e. when the value of "`mCycB`" becomes greater than 0.2 the event is activated and then when the value of `mCycB` falls under the limit 0.1, the event is fired and the variable `m` is updated with the result of the evaluation of the function `mass_div`).

In order to show the usage of templates in this small example, let us consider the degradation of X and A proteins: in the first version of the code, the spontaneous degradation mechanism has been coded for both proteins as a *delete* event, whose rate is depending on $k2p$ and $k6$ respectively. We can drop those two events and substitute the code that declares the `X` and `A` boxes with the following code:

```
template spont_degradation : pproc<<rate kin_rate>> = die(rate(kin_rate));
let X : bproc = #(x:0,X)[ spont_degradation<<k2p>> ];
let A : bproc = #(a:0,A)[ spont_degradation<<k6>> ];
```

The gain in readability, usability and maintainability of the model cannot be fully appreciated in a so small and simple model, however the general principle of recognizing common behaviors and coding them in a modular way has been used in the big model of budding yeast that is presented in Section 4.4.

Moreover, the reason why, whenever it is possible, we want to substitute the *delete* events with templates that use internal actions is because events are global rules that drive the deletion of a box from the system, but the mechanism that we would like to encode is the "local suicide" of a box that decides, from its internal code, to disappear from the system (usually after the interaction of another specific species that induces the degradation of the first one): in this way we can reduce the amount of code that has to be written because even if a protein A is degraded by different species (B, C, D, ...), the internal code of A does not change (i.e. waiting for a signal from outside on a specific interface used to sense degradation signals and then die) and the source of the signal can be any other protein of the system (with possibly different rates). Hence using templates, we can write the code that is accounting for this mechanism just once, and then we can instantiate it in different boxes just specifying the local information that differentiate each species (in our case the rate of the self degradation, or the channel on which the sensing of the degradation signal from outside is happening).

Just to conclude this section, we want to emphasize that our usage of the BlenX language is by no means exhaustive and does not take advantage of many other features of

the language that can be useful in different contexts and scenarios. For example, BlenX does not only support actions with an associated stochastic rate, but also immediate actions (using the keyword *inf* as rate value): those actions have precedence with respect to actions associated with finite rates. This can be used when, for example, more than one *change* action should be the result of a single synchonization of a box with another box: the sequence of changes can be done at infinite rate so that they will be done in zero simulation time (i.e. immediately). Other distinguishing characteristics of the language are the possibility of inserting conditional statements in the internal code of a box, coding alternative behaviors through the usage of the choice operator; defining patterns for dynamic complex formation, etc. We refer the reader to [53] to see all the distintinctive features of the BlenX language in action.

3.2 Translating a mathematical model into BlenX

In this section, we are going to show a semi-automatic method to translate existing deterministic models written with ordinary differential equations (ODEs for short) into programs written in BlenX. The simple translation of general ODE terms into a BlenX model is possible because of the expressive power of the language that allows the definition of general rate functions for the transitions. In fact, the rationale behind the translation of an ODE system into BlenX is to use the same level of abstraction adopted in the starting deterministic model so that we can define an easy mapping of ODEs into BlenX. This translation requires the usage of just a few primitives of the language.

The first step lies in converting species concentrations into species molecular number. In order to do that, it is enough to apply the $Conc = \alpha \cdot MolNum$ conversion, introducing a scalar constant α defined as $\alpha = \dfrac{1}{(N_A \cdot 10^{-6} \cdot V)}$ where N_A is the Avogadro's number, V is the volume of the modelled system expressed in ml and the initial concentration to be converted is expressed in nM. Applying the above conversion for each species S in the ODE system, we end up with a set of equation that uses the original kinetic constants, but it is written in terms of number of molecules. This allows us to use directly the manipulated ODE system's terms as stochastic rate functions, without the need of giving to each single kinetic constant its stochastic numerical counterpart.

The second algebraic modification of the ODE system that we need to do is to write explicitly the equations for the species that are appearing "implicitly" in the ODE system. If a species is present in a constant total amount but it can switch between an active and an inactive state, the time evolution of just one of the two states is usually explicitly considered, because the other (the "implicit" one) can be derived from the first. In BlenX

we have to define all the states in which a species can be, and so we need a definition also for the states not explicitly written in the initial ODEs. Hereafter, when we talk about species, we include both the original *explicit* and the added *implicit* ones.

The general mathematical description of a system of N reactant species $S_1, \ldots, S_N$ expressed in terms of number of molecules $M_1, \ldots, M_N$ and involved in $R_1, \ldots, R_R$ reactions has the following form:

$$\frac{dM_i}{dt} = \sum_{j=1}^{R_i} f_j^i(M_1, \ldots, M_N, Var_1, \ldots, Var_T) \tag{3.1}$$

where $i = 1, \ldots, N$, $R_i \leq R$, and f_j^i is a rate term referring to the amount of other species in the system $(M_1, \ldots, M_N)$ and, possibly, to $Var_1, \ldots, Var_T$ terms that represent other constant rates, discrete/continuous variables dependent from time or other complex mathematical formulation accounting for combined abstracted effect of other species of the system on the rate of change of the current species S_i (so the complete definition of the system above has to be coupled with the set of algebraic equations that entail the variation of each discrete/continuous variable of interest).

The translation from an ODE system written with the above conventions to BlenX can be summarized by the following pseudocode:

Input: the set (3.1) of ODEs;
Output: the definition of the BlenX model (i.e. the three files used as input for the BetaWB simulator).
Method:

1. expand all the possible factors of the ODEs, so that all the equations are written as summations of positive/negative terms;
2. for(i in $[1..N]$)
   for(j in $[1..R_i]$)
   define function(f_i_j) = AbsoluteValue(mathematical_expression(f_j^i))
3. for (i in $[1..T]$)
   define variable(var_i) = mathematical_expression(Var_i)
4. declare all the constants contained in the previous definitions
5. for(i in $[1..N]$) declare empty-box(S_i)
6. for(i in $[1..N]$)
   for(j in $[1..R_i]$)
   switch(kind of f_j^i reaction):
   case 'synthesis': add event "when (i : : f_i_j) new(1)";
   case 'degradation': add event "when (i : : f_i_j) delete(1)";
   case 'changing from i to k': add event "when (i : : f_i_j) split(Nil, k)";
   case 'changing from k to i': add event "when (k : : f_i_j) split(Nil, i)";
7. remove all copies but one of the replicated events that have been created for the reactions appearing in multiple ODEs
8. set the initial state of the model

Note that the choice in step (6) has to be made manually, looking at the wiring diagram of the system: this is the only step that cannot be done automatically, because of the well known *inverse problem* [60]: it is not true that a set of kinetic differential equations is induced by a unique reaction event within the class of reversible and conservative reactions. In other words, there is more than one set of reactions (i.e. structurally different networks of chemicals) that can produce the same set of ODEs. In order to see the ambiguity of the inverse problem, let us consider for example the following simple ODE system:

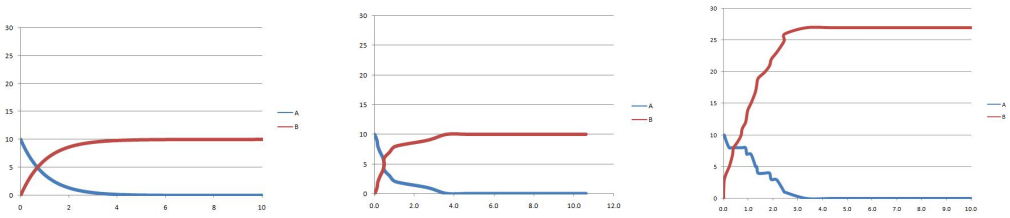
$$\frac{d|A|}{dt} = -k|A| \qquad \frac{d|B|}{dt} = k|A|$$

that can be induced by the following two alternative set of reactions:

$$A \xrightarrow{k} B \tag{3.2}$$

$$A \xrightarrow{k} \emptyset \qquad A \xrightarrow{k} A + B \tag{3.3}$$

In the deterministic solution, both systems have the same behavior. However, in a stochastic context with a small number of molecules, the two systems can show different behaviors due to the stochastic choice between the two different reactions in the second system (Fig. 3.4). In order to simulate the right set of interactions, the user must choose which is the reaction system that he/she wants to use. Usually this choice can be made by looking at the interactions depicted in the wiring diagram of the phenomena.



(a) (b) (c)

Figure 3.4: The deterministic (a) and stochastic (b-c) solutions of reactions (3.2) and (3.3), respectively, with the initial configuration of 10 A and 0 B. The stochastic simulation in (b) always reaches the end configuration of 10 B and 0 A, while the one in (c) reaches very different final configurations, due to the non-deterministic choice between the two separate reactions that describe the system.

To disambiguate between those reaction networks we need to have the knowledge of the interactions between all the species, but usually this knowledge is hidden in wiring diagrams that are not following strict or general formal conventions. Therefore also the translation that we propose cannot be a formal and general method, but it is a set of patterns that can be easily applied to each term of the ODE model. It is an useful starting point because the ODE models are usually written in such a way that the set

of inducing reactions can be recognized in a quite unique way by using the information encoded in the wiring diagram of the system.

In section 4.3 the whole translation procedure has been applied to the budding yeast cell cycle model presented in [142] in order to show that the translation is straightforward and that the stochastic simulations are, from one side, consistent with the results obtained by the solution of the ODE system, but from the other side, the stochastic model matches some characteristics that cannot be found with the deterministic one. In section 4.3 we also show how it is possible to refine the BlenX model obtained following the above translation, in order to take advantage of peculiar features of the language (e.g. templates) that allow the creation of a model that is more easily mantained and possibly composed with other models.

3.3 Issues about the usage of different abstraction levels

The presence of different levels of abstraction in the same model raises some technical issues that need to be carefully taken into account. The two main points that we want to consider here are usage of the stochastic simulation algorithm when non-elementary reactions are incorporated in the model and the (often ignored) underlying assumptions that the high-level mathematical representation of those reactions imply.

The former remark has already been discussed by several authors [3, 36, 136], and here we just want to summarize the main idea that allows us to discuss and to rely on the simulation results of the models in the following chapter of this work (where we will perform stochastic simulations of models in which elementary mass-action reactions are modelled together with higher-level complex mechanisms like Goldbeter-Koshland and Hill responses).

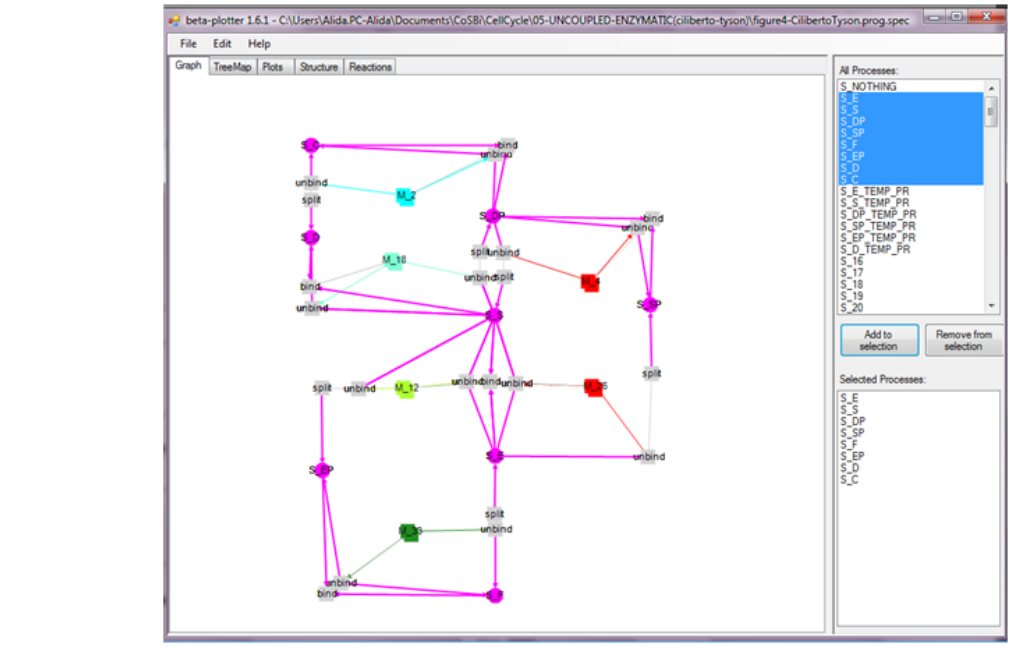
The main point is that, in general, the rate dependent functions define the reaction propensities of the stochastic model. We assume that the fundamental hypothesis of Gillespie, i.e. each reaction time is a random variable following a negative exponential distribution with rate equal to the value of the propensity function, holds for the biological system we are modelling. If this is true, a stochastic characterization of the reaction times as negatively exponentially distributed random variables is an accurate modelling choice, as proved by Gillespie [74]. In some studies considering Michaelis-Menten reactions [3], the applicability of the fundamental hypothesis was mathematically proved, and in some others dealing with the circadian rhythm this same hypothesis was experimentally verified in silico [82]. In all the cases that we are going to present in this work, because of the presence of many non-elementary reactions of many different mathematical forms, a careful validation study has been performed. In this way, even if the exactness of the

stochastic characterization of the whole mathematical model has not been verified, the validation of the simulation ouputs against the experimental measurements, both qualitatively and quantitatively, makes us confident of that the stochastic description of the system is correctly modelling the original system.

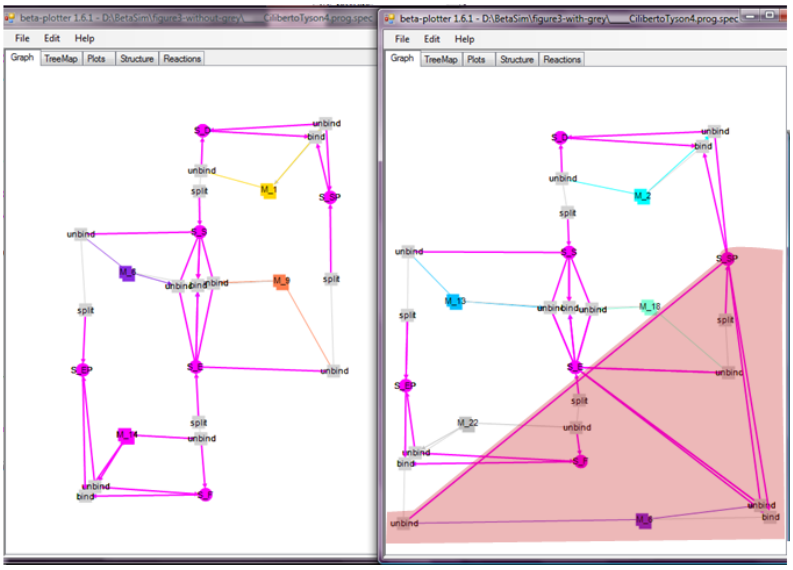
The second main problem about the presence of non-elementary reactions in a model, is the often naive attempt to unpack those complex mathematical expressions into the many elementary steps that they are abstracting. From one side, in the original model the high-abstraction level of specific mechanisms is most of the time due to the need of keeping the model manageable and/or because detailed kinetics about the individual steps are not available. From the other side, because of the philosophy that is behind the process algebra approach of modelling the single interactions between single molecules, the modeller is inclined to make a detailed model with all the elementary steps and simply look for the kinetic rates of the missing (i.e. abstracted) reactions. However often this could lead to a completely different model that has no possility of behaving like the original "packed" one, no matter how the single parameter rates are chosen.

As a main example, we refer the reader to [35] where the authors describe a very simple network of chemical species linked together with Michaelis-Menten reactions. In the paper they show that the original model (i.e. the one with the Michaelis-Menten kinetics) has a bistable behaviour (i.e. the system has two stable states, and the final outcome of the systems is determined by its initial conditions). However in the simple unpacking of the enzymatic reactions (with the explicit consideration of the intermediate complexes) the hysteresis is lost, and not just for a single parameters choice but, according to the Advanced Deficiency Theory developed by Feinberg [66, 41], the network cannot have a bistable behaviour for any assignment of positive values of the kinetic rate constants. Furthermore, the authors show that it is enough to add a background reaction between one of the (supposed to be) inactive proteins and an enzyme to re-establish the bistable behaviour of the whole system. The problem of the initial naive unpacking is in the mathematical assumptions of the "packed" version (assumptions that made possible the application of the Michaelis-Menten approximation) were hiding some mechanisms that become essential when the user is modelling the whole set of elementary steps while they are redundant if the modeller choice in this specific example is to abstract the underlying mechanism.

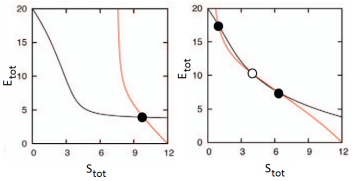
In general, those kind of abstractions could be introduced because the mathematical formulation of the system is trying to model the available experimental results directly rather than modelling the elementary molecular mechanism that result in that response: this is the reason why sometimes the mathematical rate terms have not a direct molecular



(a)



(b)



(c)

Figure 3.5: Two networks of enzymes, analyzed in [35], that model different antagonistic relationships that drive part of cell cycle. 3.5(a) shows a bistable behaviour both in the model where the Michaelis-Menten abstraction has been used and in the unpacked version. 3.5(b), instead, has a bistable behaviour only in the version of the model where a background reaction (shaded in the rightmost figure) is added. (The different plots has been produced by the Beta plotter software on a BlenX model of the three different versions of the network). 3.5(c) Nullcline of the E_{tot} and S_{tot} species (black and red curves, respectively) which represent single free species plus all the complexes in which the species appears. Left: when SP has some catalytic activity, the system is bistable. Right: when SP has no catalytic activity, the system is monostable (figure adapted from [35])

interpretation. So whenever we want to use a model in which mathematical abstractions have been used, we need to be aware of the fact that those complex mathematical expressions are hiding some assumptions (often not declared) that can be essential in the detailed version of the same model (just as a side note, not all the models presented in [35] need the background reaction in order to have the bistable behaviour: so the simple unpacking can be, in some cases, enough).

In conclusion, the usage of abstracted mechanisms is a fundamental opportunity for keeping the model as detailed as the user needs it to be (i.e. ignoring sub-parts of the model in which there is no interest) and, if a careful unpacking is performed, can be used for unraveling the undelying mechanism of the system (and/or to falsify some naive explanations of the elementary molecular interaction network).

Chapter 4

Budding Yeast Cell Cycle models

4.1 Introduction

The cell cycle is a coordinated sequence of events by which a cell replicates all its internal components and divides them into two nearly identical daughter cells. A complex molecular machinery is needed to control this crucial biological process. One of the early success stories of mathematical biology was the work done on cell cycle regulation. Through the description and analysis of the cell cycle regulatory network, theoreticians predicted several dynamical properties and unknown components of the system that were later experimentally verified. Moreover in recent years computational and theoretical approaches have been increasingly incorporated in the main stream of cell cycle research [161, 180, 93, 135, 65].

In this chapter, first we briefly describe the physiology and molecular biology of cell cycle regulation, focusing on a specific organism (i.e. the budding yeast). We present some of the most relevant mathematical models that have been studied in the literature of deterministic systems and we show how their translation into the stochastic framework presented in Chapter 3 can be used to analyze behaviours and properties that are not observable with the deterministic characterization of the system. In particular, we chose to translate different mathematical models that describe in an increasing level of details the molecular machinery that drives the growing and division of budding yeast cells. In Section 4.2 we build a stochastic model of the core molecular network behind the oscillatory behaviour of the system: we show how this small model can already be useful for a quantitative study of interesting properties, like the irreversibility of specific events. The model presented in Section 4.2 has been then refined in order to take into account more complex molecular interactions between the key cell cycle regulators: hence, in Section 4.3 we analyze a more detailed version of the same model that accounts for many experimentally verified mutants. We show that the stochastic characterization of this

version of the model is more in accordance with experimental data of partially viable mutants at the border of life and death with respect to the deterministic study that has been done in the literature. Finally, in Sections 4.4 and 4.5 we code in the stochastic framework the most comprehensive model available for the budding yeast cell cycle: we use this last refinement of the model to show how the stochastic description of the system is nicely resembling the experimental data available and can be also used for in silico analysis of single cell that just recent advance technologies made possible to be done in wet-labs.

4.1.1 Physiology of the cell cycle

Cells perform a sequence of coordinated steps (referred to as the cell cycle) that result in self reproduction. All eukaryotic cells (i.e. cells with a real nucleus) share the major processes of the cell cycle, while prokaryotes have a less characterized one. For detailed descriptions of various aspects of cell cycle modelling we refer the reader to some recent reviews [43, 45, 71, 173, 189].

The two main events in the eukaryotic process are: the proper replication of the DNA (i.e. the hereditary material of the cell) and the separation of the two copies into two daughter nuclei. Those two events take place in two different phases called S (Synthesis) and M (Mitosis) that need to occur alternately. Additionally to DNA, cells need to double all other components of the cell (proteins, ribosomes, RNAs, phospholipid bilayers, carbohydrates, metabolic machinery, etc.) in order to generate two proper offspring. Usually the doubling of the cytoplasm takes longer hence temporal gaps (G1 and G2 phases) are inserted in the cell division cycle between S-phase and M-phase in order to keep the size of the two daughter cells similar to that of the mother. This ensures that the overall cell growth cycle is coordinated with the chromosome cycle (DNA replication-division) and that maintains the homeostasis of the population.

In G1 the chromosomes are not yet replicated and the cell replication-division process is uncommitted. G1-phase can be separated into two functionally different parts and the boundary between early and late G1 is called the restriction point or "Start": it occurs when the internal and external conditions are favorable for a new round of chromosome replication and segregation. At this point a cell commits itself to the whole process. The decision is irreversible: once DNA-synthesis begins, it goes to completion and eventually the cell will finish the current cycle even if conditions are getting worse in the meantime. During S phase each DNA molecule is replicated: accuracy of those events is crucial for producing healthy and viable daughter cells, thus the synthesis is permanently checked and repair mechanisms guard the correct DNA replication. During G2 phase the cell increases its mass by duplicating its internal components. G2-phase is inserted to ensure

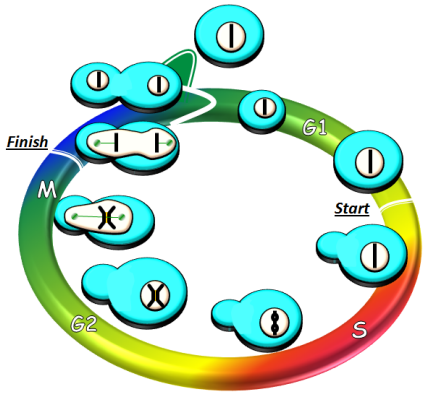


Figure 4.1: Phases of the cell cycle. The evolution in time of the budding yeast case is shown.

that DNA replication is properly finished and cells have grown to an appropriate size before mitosis.

Events during mitosis are critical for proper distribution of DNA between the two daughter cells. The mitotic process has several subphases: during prophase, replicated chromosomes condense into compact structures, in metaphase these condensed chromosomes are aligned on the center of the cell with the help of mitotic spindles. When all chromosomes are aligned the so called "Finish" transition (or meta-anaphase transition) is induced: the cohesins that hold the two sister chromatids together are destroyed allowing the chromosomes to be pulled to the opposite poles of the cell (anaphase). After distributing the DNA content in telophase the daughter nuclei form and eventually the two daughter cells separate. The two new cells are now back in G1 state and the cycle repeats (Figure 4.1).

The major steps of the cell cycle described above must be tightly regulated, thus events must be checked and corrected for errors. Evolution developed "surveillance mechanisms" that monitor progress through the cell cycle in different points in order to ensure that all the events happen in the proper order [87]. If something goes wrong, cell cycle stops at crucial *checkpoints* and waits until the problems have been repaired. Not only the state of the DNA is monitored, but checkpoints also react to the changes of the environment and ensure, for example, that cells are large enough to step into the next phase: cells must indeed grow to a critical size before committing to chromosome replication and division to guarantee the balance between the cell growth and the DNA cycle, in order to ensure that the next generation will have a size, more or less, similar to the size of the mother cell. Mistakes like DNA damage or not proper size are usually fatal if they are not repaired in single cell organism, while in multi-cellular organisms the affected cells need to be eliminated to ensure homeostatis. The development of cancer is associated with loss of cell cycle control, as tumor cells with damaged DNA do not stop replicating

and cannot be eliminated, and as a result they carry the loss of control over the next generations [102].
So summing up, the basic characteristics of cell cycle are the following:

- the alternation of DNA replication (S-phase) and mitosis (M-phase) is essential;
- growth in cellular mass is the rate limiting process, thus the DNA replication-division cycle needs to be coupled with gap phases (G1, G2) in order to slow down the progression;
- checkpoint mechanisms stop the cell cycle if any previous event has not been properly completed and ensure that repair mechanisms are initiated to try to fix the problem; and these controls are conserved in all eukaryotic organisms from yeast to human.

4.1.2 Molecular mechanism of cell cycle control

The events of the cell cycle described in the previous section are controlled by a complex regulatory network of interacting macromolecules that induce the transitions between the different phases.

CDK proteins work as effective kinases only if they are bound by a regulatory cyclin partner that helps substrate recognition. CDK/cyclin complexes initiate crucial events of the cell cycle by phosphorylating specific protein targets. For instance, CDK initiates DNA replication at the transition from G1 to S-phase by phosphorylating proteins bound to chromosomes at origins of replication (specific nucleotide sequences, where DNA replication can start), CDK also induces chromosome condensation and initiation of mitosis at the G2/M transition by phosphorylating histones (proteins involved in DNA packaging). In the opposite way, CDK inhibits the last steps of the cycle by keeping the cells in mitosis as long as CDK is active. The separation of chromosomes at the end of mitosis and cell division can occur only after CDK activity has dropped at the end of the cell cycle.

The activity of CDK/cyclin complexes can be regulated in several ways, one of which is the controlled degradation of the cyclin subunit. The Anaphase Promoting Complex (APC) with help from the regulatory protein Cdh1 labels cyclins for degradation at the end of the cell cycle. Cdh1/APC is also helped by the CDK inhibitor Sic1 that can bind to CDK/cyclin complexes and inhibit their activity. Thus G1 phase is stabilized by the concerted action of Cdh1/APC and Sic1. Interestingly, CDK/cyclin complexes can phosphorylate and inactivate Cdh1 proteins, which leads to an antagonistic relation between CDK/cyclin and Cdh1/APC. Importantly also Sic1 is in an antagonistic relation with CDK/cyclin complexes (that can induce Sic1 degradation by phosphorylating it). These antagonisms creates two, alternative, stable steady states of the control system: a G1 state, with high Cdh1/APC activity and low CDK/cyclin activity, and an S-G2-M state, with high cyclin/Cdk activity and low Cdh1/APC activity. Transitions between these

two states are facilitated by "helper" molecules that are insensitive to the antagonists. In order to better explain how the progression between the different phases is driven by this antagonistic relationship, let us consider the simple mechanical metaphor of the seesaw in Figure 4.2.

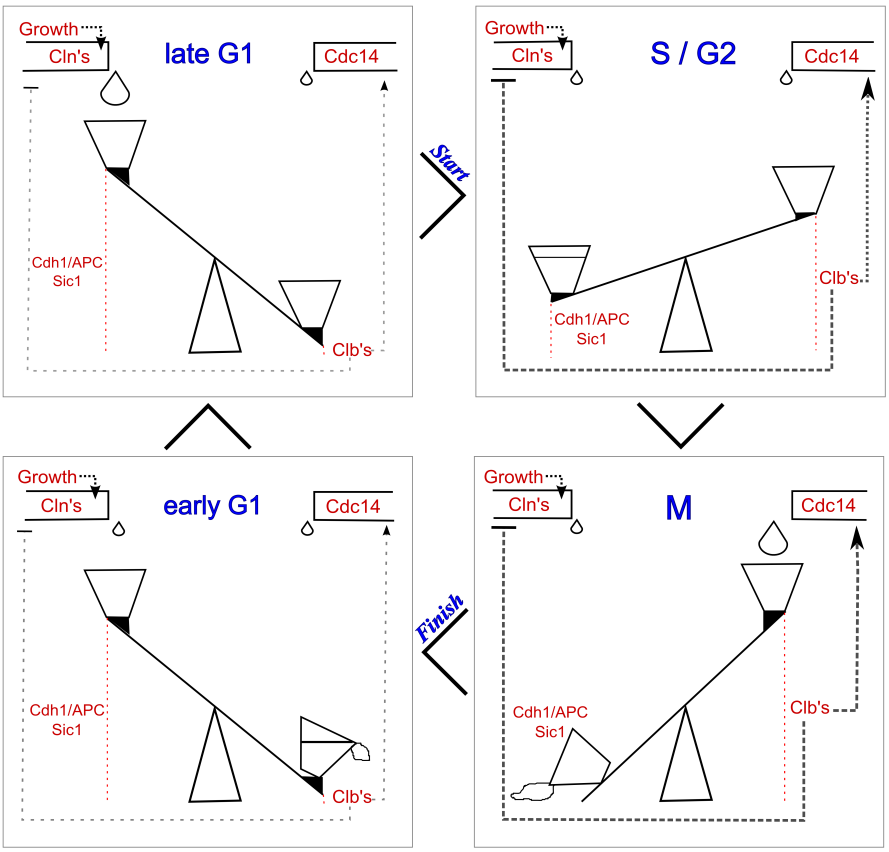


Figure 4.2: A seesaw metaphor for the antagonism between CDK/cyclin (Clb's for the specific case of the budding yeast organism) and Cdh1/APC and Sic1 proteins that drive the cell cycle oscillations (see text for details) [137]

The lever represents the strong connection between the two antagonist partners. The height of the left part of the lever represents the abundance of the Cdh1/APC protein and the stoichiometric inhibitor of CDK/cyclin (Sic1). The height of the right part of the lever represents the abundance of the CDK/cyclin complex (Clb's, for short, in the figure). The two buckets represent the helpers molecules that ease the passage between the two states. After cell division the newborn cells are in G1 phase, with high Cdh1 activity, and no CDK/cyclin complex present. As the cell grows (the "Growth" signal in the top-left corner of each sub-figure) it starts to produce cyclin (Cln's) that binds to CDK and these complexes phosphorylates Cdh1. As cell growth proceeds eventually enough CDK/cyclin

complexes are produced to inactivate Cdh1 (full left bucket in Figure "S/G2"). This leads to a further increase in CDK/cyclin level, since cyclin degradation is slowed down after Cdh1 got inactivated. This increased CDK/cyclin activity can induce DNA replication, thus the cells enter into S-phase. This abrupt increase of CDK/cyclin activity is the "Start" transition of the cell cycle. High population of active CDK/cyclin has also the effect of causing the inactivation of the transcription factors for the starter kinases Cln's, which have already accomplished their role in the cell cycle (see the emptying bucket and the small left drop in Figure "M"). CDK/cyclin also induces the synthesis of the Cdc14 protein (see the big right drop in Figure "M"). The active Cdc14 degrades CDK/cyclin complexes and activates Cdh1 (lowering the lever back on the right side). As the CDK activity reverts to low levels, the telophase completes and the cell divides. The synthesis of the APC related protein Cdc14 stops as the activity of CDK is lost (see the spilling bucket and the small right drop in Figure "early G1"). The newborn cells are back in G1 phase with low cyclin levels and the process can start again.

The result of the hysteric interaction between CDK/cyclin and Cdh1/APC can also be visualized in the oscillatory behavior of the concentrations of the different proteins as the time progresses in the cell cycle. Figure 4.3 shows the classical time courses of different key regulators of the budding yeast cell cycle.

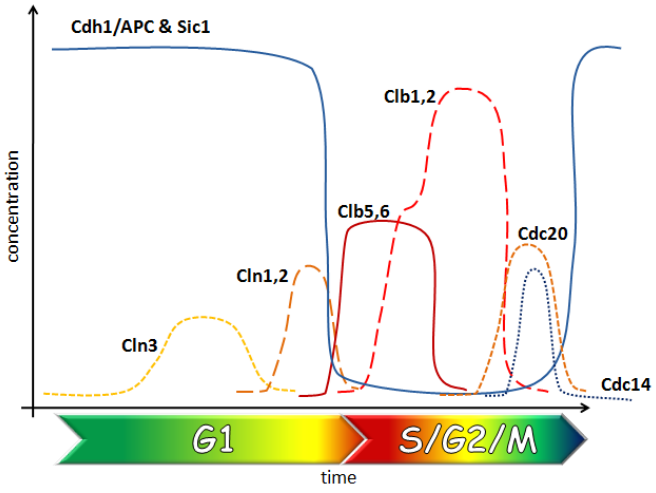


Figure 4.3: Idealized time course of the fluctuations in the activities of key regulator proteins during the cell cycle.

The dynamical properties of a cell are implicit in the topology of the protein networks that underlie cell physiology. Not only cyclin and CDK proteins are conserved among eukaryotes, but most of the cell cycle regulator proteins as well as their interactions. In this work we want to explain in more details the regulatory network of budding yeast.

4.1.3 *Saccharomyces cerevisiae* regulatory network

The cell cycle regulatory system of budding yeast is most fully worked out in [30] and the hypothetical molecular mechanism for regulating DNA synthesis, bud emergence, mitosis and cell division is depicted in Figure 4.4.

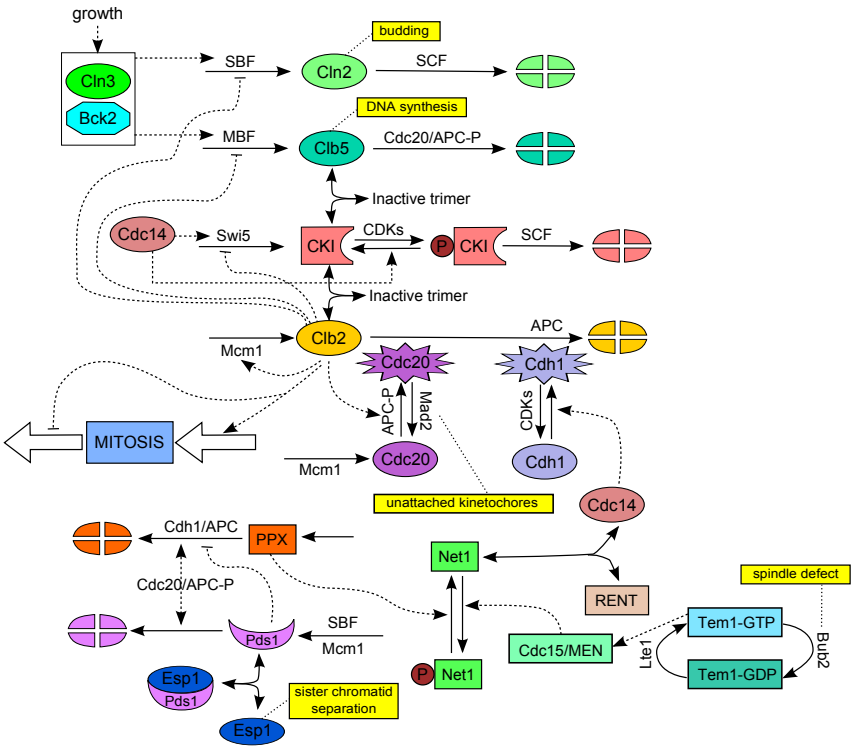


Figure 4.4: **Regulatory network of cell cycle, adapted from [30]**. In the diagram, Cln2 stands for Cln1,2, Clb5 for Clb5,6, and Clb2 for Clb1,2. Moreover the kinase protein Cdc28 is not shown explicitly because it is always in excess and it combines rapidly with the different cyclin partners as soon as they are synthesized. Newborn daughter cells must grow to a critical size to have enough Cln3 and Bck2 to activate the transcription factors SBF and MBF, which induce synthesis of Cln2 and Clb5 respectively. Cln2 is primarily responsible for bud emergence and Clb5 for initiating DNA synthesis. Clb5 activity is not immediately evident because in G1-phase cell is full of cyclin-dependent kinase inhibitors (CKI; namely, Sic1 and Cdc6). After the CKIs are phosphorylated by Cln2 and other active cyclin-dependent kinase complexes (collectively denoted by CDKs), they are rapidly degraded by SCF, releasing Clb5 to do its job. A fourth class of cyclins, denoted Clb2, are not involved in G1 because their transcription factor Mcm1 is inactive, their degradation pathway Cdh1/APC is active, and their stoichiometric inhibitors CKI are abundant. CDKs active complexes remove CKI and inactivate Cdh1, allowing Clb2 to accumulate after some delay, as it activates its own transcription factor, Mcm1. Clb2 turns off SBF and MBF. As Clb2 drives the cell into mitosis, it also sets the stage for exit from mitosis by stimulating the synthesis of Cdc20 and by phosphorylating components of the APC. Meanwhile, Cdc20/APC is kept inactive by the Mad2-dependent checkpoint signal responsive to unattached chromosomes. When the replicated chromosomes are attached, active Cdc20/APC initiates mitotic exit. First, it degrades Pds1, releasing Esp1, a protease involved in sister chromatid separation. It also degrades Clb5 and partially Clb2, lowering their strenght on Cdh1 inactivation. In this model, Cdc20/APC promotes degradation of a phosphatase (PPX) that has been keeping Net1 in its unphosphorylated form, which binds with Cdc14. As the attached chromosomes are properly aligned on the metaphase spindle, Tem1 is activated, which in turn activates Cdc15. When Net1 gets phosphorylated by Cdc15, it releases its hold on Cdc14. Cdc14 (a phosphatase) then does battle against the cyclin-dependent kinases: activating Cdh1, stabilizing CKIs, and activating Swi5 (the transcription factor for CKIs). In this manner, Cdc14 returns the cell to G1 phase (no cyclins, abundant CKIs, and active Cdh1)

In general, in order to determine whether an hypothetic regulatory network is correct, the classical approach is to convert the mechanism informally described by the wiring diagram into a formal mathematical/computational model and to compare the results of those models to the observed behavior of the chemical reaction system. If the results fit the observations, then the mechanism is provisionally confirmed (pending further experimental investigations). If not, inconsistencies identify aspects of the mechanism that require revision and further testing. Although a mechanism can be disproved if it is inconsistent with well-established facts, it can never be proved correct, because new observations may force modifications and additions. Hence, the hypothetical interaction network presented in [30] and portrayed in Figure 4.4 cannot be considered "true" but it is the most complete, validated and detailed model that reasonably approximates what is going on inside yeast cells.

The wiring diagram in Figure 4.4 implies a set of dynamical relationships among its components, a quite complex network with several intertwined feedback loops. A detailed characterization of such a complex system can be achieved only by mathematical and computational approaches that describe the temporal and spatial evolution of the system.

The type of equations/languages to be used depends on the biological questions under consideration. For instance, genetic regulatory circuits might be modelled by differential equations or by Boolean networks. Spatial signalling might be modelled by partial differential equations or by cellular automata. When small numbers of molecules are involved, stochastic models must be used. And in the plethora of languages that support stochastic simulations, the choice should be driven by the different user needs (i.e. an easy/automatic translation from a language to another, a final model written in a standard language supported by many software tools, a final model that is easily composable with other models, fast simulations results, different post-processing of the simulation results, ...). We refer the reader to Section 2.3 for a detailed discussion of the different choices that have been made in the literature for modelling budding yeast cell cycle and other biological systems, both with mathematical and computational approaches. Here we just summarize the main analyses that have been done on this organism and that inspire and guide the in-depth examination – presented in the rest of this chapter – that we have done on this interesting case study.

As some data on the key regulator of cell cycle (CDK) were found by Nurse, Hartwell and Hunt, theoreticians started to create models to understand how CDK can regulate cell cycle events [79, 181, 186]. Most of these works focus on the basic properties of cell cycle regulation. The two landmark papers by Katherine C. Chen and colleagues from the Tyson lab stand out as the most influential mathematical models of cell cycle regulation [30, 31].

These models are based on an extensive literature data collection on more than 100 budding yeast cell cycle mutants and the final model can simulate the findings of almost all these experiments. A comprehensive website reports all these results, together with detailed description of the model (`http://mpf.biol.vt.edu/research/budding_yeast_model/pp/`). In 2002 Fredrick R. Cross and his group performed experiments following the suggestions made in the early paper by Chen and colleagues and confirmed the predictions ([42]). In this groundbreaking paper Cross went much further and made an extensive experimental test of various properties of the model while also performing some crucial measurements to help further model development. Two years later the measurements and corrections by Cross were implemented into a new extended version of the Chen model [30]. The resulting model was tested in the deterministic framework against the behavior of 131 mutants and marginally failed only on 11 of them. Considering the fact that many cell cycle processes are driven by proteins which are present in low concentration [101], the few inconsistencies between the model and experiments can be the result of the noise acting on those regulatory proteins. Moreover intrinsic molecular fluctuations are also probably partially responsible for the fluctuation in cell cycle properties like the size at division or the length of the different phases of the process.

4.2 Irreversibility study

Mathematical models have been investigating the dynamics of the interactions that drive the "Start" transition of the cell cycle (i.e. the abrupt increase of Cdk/CycB activity after the inactivation of Cdh1) [31, 141]. It has been proposed that the positive feedback loop that is the result of this antagonism between Cdk/CycB and Cdh1/APC can create bistability and hysteresis in the system [31]. Experiments proved the existence of bistability in the cell cycle of budding yeast cells [42], but the irreversibility of this transition was never tested yet.

We decided to analyse this module by applying both stochastic simulations analysis in BlenX and probabilistic model checking [38] on a simple budding yeast cell cycle model [6]. Model checking allows one to state and verify relevant properties of a system. Algorithms and software tools exist that take as inputs both the model M and a formula ϕ (expressing, in temporal logic, the property that needs to be verified) and return either a positive answer, if ϕ is satisfied by M (denoted $M \models \phi$) or a negative one if that is not the case (denoted $M \not\models \phi$). The peculiarity of model checking is that verification of ϕ against M is achieved through an exhaustive exploration of the model state space, hence the outcome of model verification is *exact*, as opposed to the *approximated* results obtained through model simulation. The obvious downside of model checking, is that,

due to the complexity of many real systems, the resulting model dimension blows up to the point that model checking becomes infeasible. So, we chose to rely on probabilitistic model checking for studying the smaller model of the core of the cell cycle because the quantification of the probability of the irreversibility of the "Start" transition can be done in an exact way but since this approach cannot handle more complex models, we show that analysis of a bigger BlenX model can give similar insights on the system (even if just approximated).

The core mechanism that we investigated with the model checking approach is depicted in Figure 4.5.

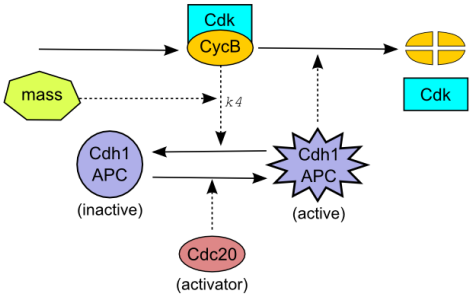


Figure 4.5: **Antagonism between Cdk/CycB complex and the Cdh1/APC.** Cdh1/APC (denoted Y in equations (4.1-4.3)) induces the degradation of cyclin while Cdk/CycB (denoted X in equations (4.1-4.3)) inactivated Cdh1/APC by phosphorylation. Increase in cell mass helps to concentrate Cdk/CycB into the nucleus, where it acts on Cdh1/APC, thus following [142] we assume that increase in cell size elevates Cdk/CycB's efficiency to inactivate Cdh1/APC. The model is similar to the one shown in Section 3.2 apart from the fact that the activator protein here is considered constant. The parameter that is the subject of this study (i.e. k_4) is also shown in the picture.

As previously illustrated, the antagonistic interaction between Cdk/CycB and Cdh1/APC stands at the core of cell cycle regulation. This switch makes the decisions on commitment to start the cell cycle [30]. The wiring diagram of Figure 4.5 was turned into ODEs by Novak and Tyson [142]. The dynamic of the real valued variables X and Y are described by the following ordinary differential equations, taken from [142]:

$$\frac{dX}{dt} = k_1 - (k_2' + k_2'' \cdot Y)X \tag{4.1}$$

$$\frac{dY}{dt} = \frac{(k_3' + k_3'' \cdot A)(1 - Y)}{J_3 + (1 - Y)} - \frac{k_4 \cdot m \cdot X \cdot Y}{J_4 + Y} \tag{4.2}$$

$$\frac{dm}{dt} = \mu \cdot m \left(1 - \frac{m}{m_*}\right) \tag{4.3}$$

Note that a unitary (constant) total concentration of Cdh1/APC complex is assumed, thus $(1 - Y)$ represents the inactive amount of Cdh1/APC. Furthermore, A indicates the amount of the *activator* (Cdc20) protein which influences activation of Cdh1/APC. For simplicity, in this model, we assume A constant. Finally m denotes the cell mass and its dynamic behavior is driven by equation (4.3), where μ is the growing factor and m_* is the maximum level that the mass can reach.

In order to see the effect of $k4$ on the "Start" transition and to quantify the loss of irreversibility that lowering the strength of this reaction causes, we decide use probabilistic/stochastic model checking [89, 134]. Continuous Time Markov Chains are a well established form of discrete-state stochastic processes largely used for modelling and analysis of many different types of systems. A CTMC model can be thought of as a graph whose states correspond to variables' value and whose transitions indicate the dynamic of the modelled system. In a CTMC, transitions are labelled with real valued numbers, representing the rate of an exponentially distributed delay (the time consumed by the transition to take place). Once a CTMC model is developed then it can be analysed in several manners. Classical steady-state and transient analysis, provides information about the system evolution, respectively, *in the long run* (steady-state), or with respect to a specific instant of time (transient analysis). If the model is too large, stochastic simulations can be applied to derive relevant statistics. For a detailed description of CTMC models we refer the reader to [178].

Starting from ODEs (4.1-4.3) a CTMC model of the Cdk/CycB-Cdh1/APC module has been derived and coded into the PRISM probabilistic model checker [110], a software tool implementing different CTMC analysis. The coding into PRISM firstly requires the discretization step obtained through the conversion mechanism of concentration into molecular numbers similar to what we have done for coding the models in BlenX. Moreover, after the conversion, in order to limit the state-space explosion of the CTMC model, the initial values have been scaled by an order of magnitude, hence in the final model we have $X_{init} = 0$, $Y_{init} = 42$ (note that reaction rates need to be re-scaled accordingly)

Noise sensitive model. The CTMC model has been provided with means to keep track of the noise level. Noise can be thought of as a fluctuation, within a given threshold, around the current level of signal (i.e. the level of molecules). Keeping track of signal noise allows one to account for "noise-free" variations in the level of molecules of a given species[1]. For that purpose the CTMC model has been equipped with a parameter $noise\_d$ which can be set to the desired level of noise. As a result, a sequence of transitions in the CTMC model is recorded as an actual increasing/decreasing path only when it consists of

[1] in fact noise fluctuations are not relevant for the sake of our analysis and should be disregarded.

at least *noise_d* consecutive increasing/decreasing transitions (see Figure 4.6). In practical terms this is achieved by means of a specific coding in the PRISM language so that (noise-free) monotonic trends for species X and Y can be straightforwardly captured by means of temporal logic formulae such as (4.5) and (4.4).

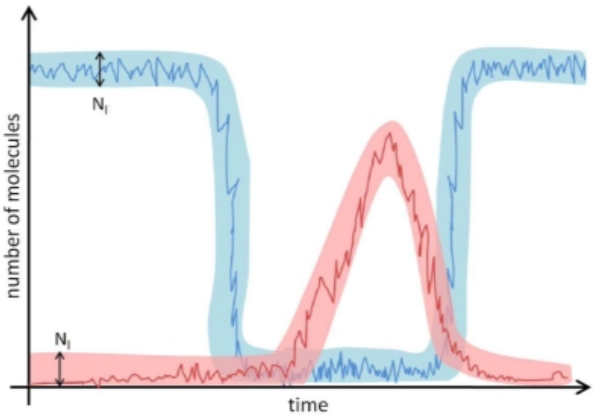


Figure 4.6: Visual representation of noise-free oscillations of Cdk/CycB and Cdh1/APC. Only fluctuations out of the coloured band are considered actual fluctuations and oscillations inside the band are considered just noise.

The irreversibility nature of the interactions between Cdk/CycB and Cdh1/APC can be formally expressed using the *Continuous Stochastic Logic* (CSL) [4, 5], that is a language for stating properties referring to CTMC models. A full description of CSL formulae is out of the scope of this work but the general idea is that the language allows us to express and verify basic properties of a system like *"after time t, the number of Cdk/CycB molecules is below n until the cell mass is below m"* (a more detailed description of CSL and references can be found in Appendix A.3).

Irreversibility in the Cdk/CycB-Cdh1/APC module of the cell-cycle corresponds to the monotonic trend of both X (Cdk/CycB) and Y (Cdh1/APC). In the initial state X is low ($X = 0$) whereas Y is high ($Y = 42$), when the cell's mass reaches a certain threshold then Y gets inactivated thus X starts growing. We formally characterise monotonicity of Y and X with the following CSL path formulae:

Probability of Monotonic increase of Cdk/CycB. *'What is the probability that the number of molecules of Cdk/CycB increases monotonically until the value v is reached?"*

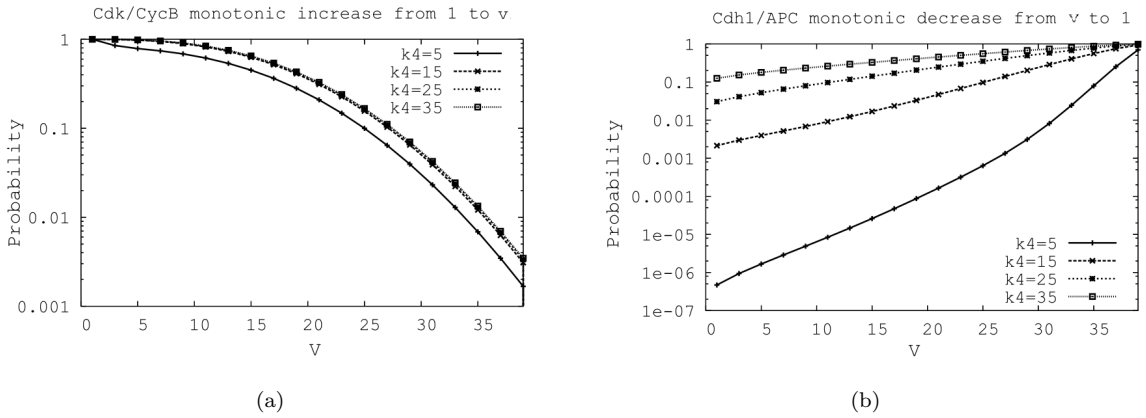
$$P_{=?}[(increase\_X\ U\ (X = v)] \tag{4.4}$$

Probability of Monotonic decrease of Cdh1/APC. *"What is the probability that the number of molecules of Cdh1/APC decreases monotonically until the value v*

is reached?"

$$P_{=?}[(decreasing\_Y\ U\ (Y = v)] \tag{4.5}$$

Verification of such formulae through the PRISM tool quantifies the likelihood of monotonicity (hence irreversibility) to be maintained (for X and Y) up until the value of v (which can be made to vary in $[0, 42]$). Figure 4.7(a) and Figure 4.7(b) show the results of model checking verification for formulae (4.5) and (4.4), respectively, as a function of the reaction rate k_4, which is one of the key parameters governing the interaction between X and Y (see Figure 4.10). Generally speaking results depicted in Figure 4.7(a) and Figure 4.7(b) show, as expected, that decreasing the influence of the Cdk/CycB dimer on the inactivation of Cdh1/APC (i.e. lowering of the rate k_4) decreases the likelihood of a monotonic trend (thus of irreversibility) of both X and Y. Furthermore, by comparing Figure 4.7(b) and Figure 4.7(a), we observe the existence of a slight asymmetry between the monotonicity of Y and X, which is: it is more likely for X to span upwards the whole interval $[0, 42]$, than for Y to span downwards $[42, 0]$.



(a) (b)

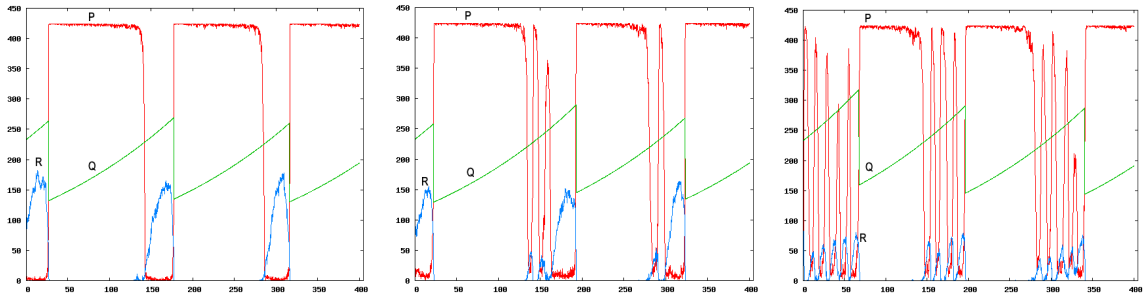
Figure 4.7: Monotonic increase of Cdk/CycB (decrease of Cdh1/APC) with cell mass growing form the initial value of 50 up to 100.

In order to confirm the results obtained through probabilistic model checking we performed stochastic simulations of a more detailed version of the cell cycle regulatory network. By means of the BetaWB (see Chapter 3) we developed a model of the wild type network [142] as depicted in Figure 4.10. For more details about this model (how we coded it and which peculiarities it has) we refer the reader to Section 4.3. Here we just want to illustrate the results connected to the irreversibility study and obtained from the Multiple stochastic simulative Replications in Parallel (*MRiP*) that we perform on this model.

MRiP approaches are frequently used to speed up simulations by working out inde-

pendent replications of the same stochastic trajectory on multiple computers. Each run is calculated starting from a *seed* chosen among a stream of pseudorandom numbers obtained with the *leap-frog* technique [59] by splitting linear congruential generators. Such a seed guarantees that the resulting trajectories are approximately uncorrelated. So doing, more observations can be collected during a given time interval than running a single replication on one computer within the same period of time [64, 76, 54].

We built 8 identical models except for the kinetic parameter k_4, that accounts for the inactivation of Cdh1/APC by Cdk/CycB and that we systematically decreased from 35 to 0, step 5. Therefore, to guarantee the trustworthiness and the statistical accuracy of the following analyses, we ran a batch of 100 simulations for each new parameter value to the number of 800 simulations. In Figure 4.8, we show three sample simulations with decreasing k_4 parameter. The simulations with the original set of parameters (Figure 4.8(a)) is reproducing the solutions of the original ODE model in [142], apart from the stochastic noise. At a first glance, it is evident that even a small change of the parameter makes less stable the supposed irreversible Cdh1 decreasing activity (curve P in each graph). Such activity results in quick and sustained oscillations of high amplitude waves.



(a) (b) (c)

Figure 4.8: Sample simulations of the wild type model depicted in Figure 4.10. Each of the plots shows 3 curves: (P) Cdh1/APC, (Q) mass, (R) Cdk/CycB. All the simulations have been equally sampled and then 2000 points (a point every 0.2 seconds) have been plotted. 4.8(a) plot corresponds to the original model ($k_4 = 35$), 4.8(b) plot comes from a model with $k_4 = 15$ and 4.8(c) plot is related to a model with $k_4 = 5$.. Cell division event follows the rule illustrated for the model in Section 4.3: the mass value is halved when the concentration of active Cdk/CycB falls below an assigned threshold (0.1, as in the original model) after having raised above another threshold (here 0.2).

Moreover, Figure 4.8(c) shows that the more k_4 is decreased, the less stable is the phase transition, i.e. an increasing number of oscillations of the Cdh1 concentration are present in each cell cycle.

In this context, in order to figure out how sensitive the irreversibility of the Cdh1/APC inactivation is to the parameter of interest, we statistically inspected the 800 simulations

results, counting the number of oscillations observed during a complete cycle. The results depicted in Figure 4.9 show the percentage of cases with *only one* Cdh1/APC inactivation per cell cycle. They show that the probability of an irreversible behavior is decreasing, as the parameter is decreased.

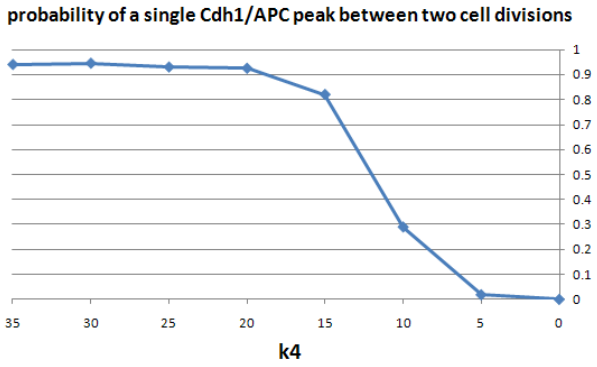


Figure 4.9: Probability of a single Cdh1/APC peak between two cell divisions

Our results provide a quantitative measure for the irreversibility of the "Start" transition of the cell cycle, reached by probabilistic model checking of the Cdk/CycB - Cdh1/APC core module of cell cycle regulation. We show that by weakening the strength of the positive feedback loop (by reducing k_4) the irreversibility gets lost. With stochastic simulations of the budding yeast cell cycle model that includes also the auxiliary regulators of this module, we show that indeed the above mentioned parameter variations can perturb the irreversibility of the "Start" transition of the cell cycle.

4.3 Viability study

In this section we applied the semi-automatic procedure, explained in Section 3.2, that allows us to translate an existing model written with ODEs into BlenX. The whole process is applied to the budding yeast cell cycle model in [142], in order to show that the stochastic simulations are, from one side, consistent with the results obtained by the solution of the ODE system, but from the other side, the stochastic model is able to explain some peculiar behavior of some mutants that cannot be captured by the deterministic one.

We will focus hereafter on the biochemical machinery that controls Cdks activity that has been modeled with ODEs by Novak and Tyson (see [142], pag. 270), and that is depicted in Figure 4.10.

In order to see the procedure of Section 3.2 in action, we do not report here the 8 equations composing the model in [142], but only the ODEs of two species that contain all the basic features needed to show the translation in BlenX of the model (see equations

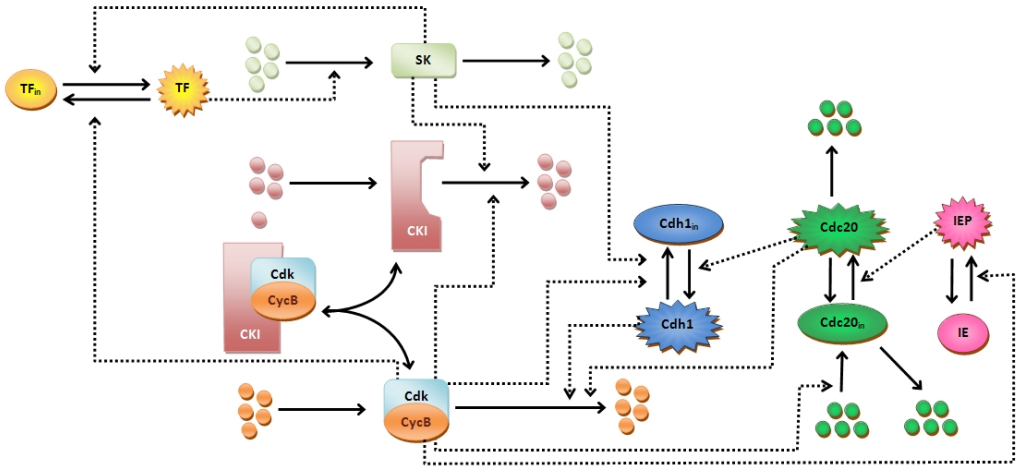


Figure 4.10: Graphical representation of cell cycle engine. Solid lines link reactants to products, dashed lines represent the mediation effect that some species have on reactions. Model taken from [142]

in Figure 4.11 below).

$$\frac{d|CDC20\_IN|}{dt} = \underbrace{k5p/\alpha}_{synthesis} + \underbrace{\frac{(k5s/\alpha)\cdot(\alpha\cdot m\cdot CycB)^n}{(J5)^n+(\alpha\cdot m\cdot CycB)^n}}_{(induced)\ synthesis} - \underbrace{k6\cdot|CDC20\_IN|}_{degradation} + \\ + \underbrace{\frac{k8\cdot|CDC20\_A|}{J8+\alpha\cdot|CDC20\_A|}}_{inactivation} - \underbrace{\frac{k7\cdot\alpha\cdot|IEP|\cdot|CDC20\_IN|}{J7+\alpha\cdot|CDC20\_IN|}}_{activation} \tag{4.6}$$

$$\frac{d|CDC20\_A|}{dt} = \frac{k7\cdot\alpha\cdot|IEP|\cdot|CDC20\_IN|}{J7+\alpha\cdot|CDC20\_IN|} - \frac{k8\cdot|CDC20\_A|}{J8+\alpha\cdot|CDC20\_A|} - k6\cdot|CDC20\_A| \tag{4.7}$$

$$\frac{dm}{dt} = \mu\cdot m\cdot(1-m/mstar) \tag{4.8}$$

Figure 4.11: Equations for the activation/inactivation of the Cdc20 protein and the rate law for the growing of the mass. $|S|$ is the number of molecules of species S and the different ks (and Js) are the deterministic kinetic parameters. The part of the model above includes the reference to a discrete variable ($CycB$) that represents the activity of the dimer Cdk/CycB. $CycB$ is calculated with an algebraic expression which can be found in [142] with the complete set of equations and variables. The α conversion factor has been calculated using the (average) volume of the budding yeast cell nucleus (which is assumed to be $V = 0.7043188\cdot10^{-15}$, corresponding to roughly 1.67% of an initial average cell volume of $42fl$).

The $CDC20\_IN$ species contains a positive term for the rate of its **synthesis** ($k5p/\alpha$) and a negative term for the rate of its **degradation** ($k6\cdot|CDC20\_IN|$). The result of the codification of the rate functions definition is in Figure 4.12(a).

The next step is the encoding of the structure of the model. We define an empty box for the species $CDC20\_IN$ and then we add a *new* event that represents its synthesis and a *delete* event that represents its degradation. The rates of the events are the functions

(a)
```
let k5p : const = 0.005; let k6 : const = 0.1;      //step 4
let alpha : const = 0.00302023;                      //step 4
let d_dtCDC20_IN_1 : function = k5p/alpha;           //step 2
let d_dtCDC20_IN_5 : function = k6*|CDC20_IN|;       //step 2
```

(b)
```
let CDC20_IN : bproc = #(a,CDC20_IN)[ nil ];         //step 5
when(CDC20_IN : : d_dtCDC20_IN_1 ) new(1);           //step 6
when(CDC20_IN : : d_dtCDC20_IN_5 ) delete(1);        //step 6
```

(c)
```
let d_dtCDC20_IN_3 : function =                      //step 2
 (k8*|CDC20_A|)/(J8+alpha*|CDC20_A|);
let d_dtCDC20_IN_4 : function =                      //step 2
 (k7*alpha*|IEP|*|CDC20_IN|)/(J7+alpha*|CDC20_IN|);
```

(d)
```
let CDC20_A : bproc = #(a,CDC20_A)[ nil ];           //step 5
when(CDC20_IN::d_dtCDC20_IN_4) split(Nil,CDC20_A);   //step 6
when(CDC20_A::d_dtCDC20_IN_4) split(Nil,CDC20_IN);   //step 6
```

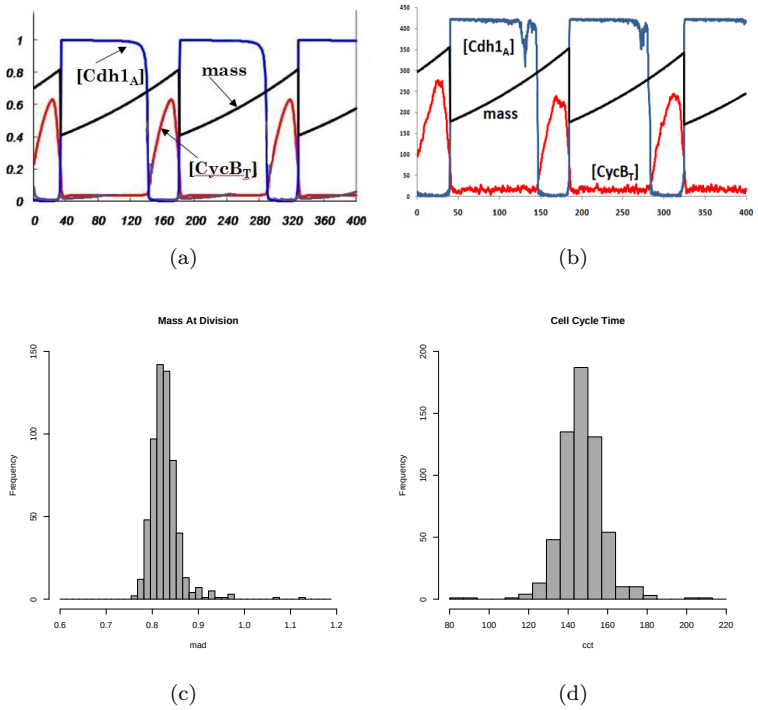
Figure 4.12: BlenX code showing the result of the steps of the translation method described in Section 3.2 on the specific equations in Figure 4.11: synthesis/degradation (a) and activation/inactivation (c) rate functions, synthesis/degradation (b) and activation/inactivation mechanism (d).

defined in Figure 4.12(a) and structure of the model is in Figure 4.12(b). The $CDC20\_IN$ species contains a positive Michaelis-Menten term that is representing the **inactivation** of the $CDC20\_A$ and a negative Michaelis-Menten term that is representing the **activation** of the $CDC20\_IN$. Coding this rate function terms similarly to the previous case, we obtain the code in Figure 4.12(c). In the model we just have to add the box for the CDC20_A species because we have already defined the box for the inactive species. Then we add two events, one encoding the inactivation of the active Cdc20 and one encoding the activation of the inactive Cdc20. Their rate are the ones in Figure 4.12(c) and the code for this part of the model is in Figure 4.12(d). All the terms of the complete ODE system can be seen as one of the previous cases, so even if they are driven by complex kinetics (as the Hill function in the synthesis of $CDC20\_IN$) they can be easily coded by a *new/delete/split* event.

With the rules introduced so far we encode almost the whole set of ODEs from [142]. The only one left is ODE (4.8), which is not representing a chemical species but the mass of the cell. It can be seen as a variable that has to be updated, following the function in equation (4.8), with discrete time steps. With BlenX is possible to define those kind of variables simply copying their ODE with the other rate functions, declaring it as a *var* rather than a *function*. Also cell division is an event that is not explicitly

coded in the ODE system, and it halves the mass value when the concentration of active Cdk/CycB falls below an assigned threshold (0.1, as in the original model) after having raised above another threshold (here 0.2). In BlenX we can simply add the event "`when(:mCycB→0.2,mCycB←0.1:)update(m,mass_div);`", which tells the simulator to trace the state of the variable $mCycB$ and whenever this variable overcomes the 0.2 threshold and then goes back under the 0.1 threshold, the event that updates the value of the mass with its halved value is executed. With this last rule we can complete the translation of the model from ODE to BlenX.

Following [135], we compare the results provided by the BlenX model and the deterministic one for the wild type of budding yeast. The model simulation was performed with BetaWB and the only difference among the runs of the model is the stochastic fluctuation (Figure 4.13(a)-4.13(b)).



(a) (b)

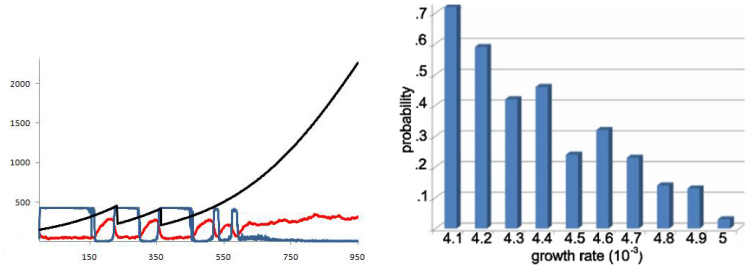
(c) (d)

Figure 4.13: Deterministic (4.13(a)) and stochastic (4.13(b)) results for the wild type model. (4.13(c),4.13(d)) Statistics of mass at division and cell cycle time (respectively) on 20 runs of the stochastic model. The means of the stochastic values are consistent with the results of the deterministic model (mean mad = 0.826989, mean cct = 146.8167); however deviation from the mean values can be seen and, in particular for the mass at division value, the bell-shaped distribution unbalanced on its right side is caused by some outlier cycles that divides with a bigger mass and are able to influence in the same direction some of their following offspring cycles ($\text{cv}_{mad} = 3.88$, $\text{skewness}_{mad} = 3.07$, $\text{cv}_{cct} = 7.51$, $\text{skewness}_{cct} = 0.037$)

All the results and statistics deduced from the stochastic simulations are in agreement with the deterministic model of those mutants (see Figure 4.13(c)-4.13(d)).

In order to show the insights that can be obtained with the stochastic model, we

show results for the mutant obtained with the deletion of the destruction box of cyclin Clb2 (which reduces its degradation rates), and with the deletion of cyclin Clb5 (which reduces the overall CycB level). This model can be obtained changing just some specific parameters of the original model. Experimental results show that this mutant is viable, but only with a decreased growth rate [42]. Therefore we reduce the rate constant of growth from the default value 0.005 min^{-1} to 0.0041 min^{-1}. Both the deterministic and the stochastic model results correctly show its viability. It is interesting to observe that the deterministic model is able to fit the lethality of the mutation with growth rate 0.005 and its viability with growth rate 0.0041, but cannot predict the intermediate situations: it is instead reasonable to expect a continuous transitions as the growth rate varies in the interval [0.0041-0.005], with some mutant cells that have a limited survivability for values of the growth rate inside the interval and a death probability increasing approaching the lethal situation of 0.005.



(a) (b)

Figure 4.14: Stochastic run for a nutritional sensitive mutant (Clb2dbΔ clb5Δ). This peculiar mutant is viable only under circumstances that slow down its growth rate. It is interesting to observe that the deterministic model is just able to fit the lethality of the mutation in glucose (growth rate $\mu = 0.005$), but actually, the transition from dead to viable for the ODE model is at $\mu = 0.0041$, and the model cannot predict any intermediate situation. From the plot of the stochastic version of the model (with growth rate 0.0045) (Figure 4.14(a)) we can instead see that the cell was able to generate 2 offspring generations before losing the ability of surviving further, so intermediate behaviour can be identified and quantified (4.14(b)). We conducted an in-silico experiment to evaluate the probability that the progeny of a single mutant cell would be able to divide at least 10 times before dying (forming a small colony), with growth rate within the interval [0.0041-0.005].

Small colonies of mutants cells with growth rate in this interval have been experimentally observed [42], so we are expecting that if a mutant is able to complete a sufficient number of cells cycles before dying, a colony may develop, even if its overall growth would be slow. We conducted an in-silico experiment to evaluate the probability that a single mutant cell would be able to generate at least 10 offspring generations before dying, varying the growth rate within the interval [0.0041-0.005] (we ran 100 simulations for each value). The results in Figure 4.14 clearly show that colonies of the mutant may exist for values of the growth rate higher than the threshold which sets the upper limit for the viability of the mutant in the ODE model. We can conclude that stochastic simulations are important to check the partial viability of mutants at the border of life and death.

4.4 Stochastic characterization of Chen model

In this section we show the statistics of the stochastic version of the model in [30] whose biological behavior as already been explained in Section 4.1.3. The chemical reaction network that has been translated into BlenX is the one depicted in Figure 4.4.

The whole set of ODEs relative to this reaction network can be translated into BlenX just following the rules illustrated in the previous sections, but because of the size of the model, this can lead to a model that it is difficult to maintain and possibly integrated in a bigger model accounting for other regulatory mechanisms of the cell. This is the reason why, to produce the code shown in Appendix A.1, we refined the initial model using some more advanced BlenX features to code this biological network. In particular we take advantage of the usage of templates (described in Section 3.2) and we code specific internal behavior for different species so that intercommunications, conditional and alternative execution of internal process can encode the specific properties of each species (see Figure 4.15).

```
let CDC20  : bproc = #(cdc20_degr, CDC20_ty), #(cdc20_out, CDC20_out_ty),
                     #h(cdc20_act, CDC20_ACT_ty)
                     [ Cdc20_proc ];
let CDC20i : bproc = #(cdc20_degr, CDC20I_ty), #(cdc20_out, CDC20_out_ty),
                     #(cdc20_act, CDC20_ACT_ty)
                     [ Cdc20_proc ];
```

```
let Cdc20_proc : pproc = receiving_degr_signal<<cdc20_degr>> |
                         spont_degradation<<kd20>> |
                         self_activation | rep rec?().self_activation |
                         send_degr_signal | rep rec2?().send_degr_signal |
                         activation_by_iep | rep rec3?().activation_by_iep;
```

```
template spont_degradation : pproc <<rate kin_rate>> = die(rate(kin_rate));
template receiving_degr_signal : pproc <<name channel>> = channel?().die;

let send_degr_signal : pproc =
        if(cdc20_degr, CDC20_ty) then cdc20_out!().rec2!().nil endif;
let self_activation : pproc = if(cdc20_degr, CDC20I_ty) then
        ch(rate(ka20_p),cdc20_degr,CDC20_ty).hide(cdc20_act).rec!().nil endif;
let activation_by_iep : pproc = if(cdc20_degr,CDC20I_ty) then
        cdc20_act?().ch(cdc20_degr,CDC20_ty).hide(cdc20_act).rec3!().nil endif;
```

Figure 4.15: BlenX code showing the usage of *templates* that allow the model to be more easily integrated in a compositional manner. In particular the three extracts above show how to code different behaviors that the Cdc20 protein can perform.
In the first part, the definitions of the two states of the Cdc20 protein (i.e. active (*CDC20*) and inactive (*CDC20i*)) are shown. The two boxes differs only with respect to the state of the interfaces, while the internal behavior is coded by the same bio-process (*Cdc20_proc*) that, in its own definition, will differentiate the behavior of the species according to state of the box. The second part contains the definition of the internal process of the protein. In particular the actions that it can perform has been made available in parallel and, at each simulation step, one of those actions will be chosen (and fired) according to their rate in the context of the SSA algorithm. (Caption continued on the next page)

The final BlenX code of the model can be found in Appendix A.1 and it can be used as input of the BetaWB simulator to reproduce the results presented in this work.

At the web site [19], full details about the deterministic model and the solutions of the different mutants can be found. Here the authors present a set of 131 mutants built using exactly the same equations and parameter values of the wild type except for those parameter changes that are dictated by the nature of the mutation. Some of those mutants are viable, while some others are not. The viability of a cycle is not determined just by the fact that all the proteins are showing an oscillatory behavior in time, but also the timing of the different events that characterize the passage between a phase and the following one must be taken carefully into account. In order to be able to tell if the BlenX output are reproducing viable/inviable cycles, we produced a small program that analyzes the stochastic traces produced by the BetaWB simulator and tell, according to the rules in [146, 2], if a cycle has to be considered alive or not and (if not) what is the reason/stage of its death. The logical rules implemented are illustrated in Figure 4.16.

We performed multiple stochastic simulations in parallel for the wild type parameter setting of the model: in Figure 4.17 we show some of the statistics that has been deduced from the output of these simulations.

The oscillations are consistent with the results of the solution of the ODEs with an evident effect of the noise on the variance of the different properties of the cell cycle.

We performed a complete study (similar to the one described above for the wild type model) for all the mutants listed on the web site [19]. The results for all the viable

Figure 4.15: (continued) The behavior that the protein can perfom are:
a) waiting on one of its own channel interfaces (*cdc20_degr*) for signal of degradation (implemented by the *receiving_degr_signal* bio-process – whose definition is in the bottom part of the figure – that just waits for a signal on a channel passed as parameter and then performs the *die* action that deletes the box from the system);
b) start a spontaneous degradation at rate *kd20* (implemened by the *spont_degradation* bio-process – whose definition is in the bottom part of the figure – that just performs a *die* action with the rate set as a parameter);
c) the *self_activation* process (made always available to the box by the replication process guarded by the input/output on the *rec* channel) which is responsible for changing the states of the interfaces from the inactive to the active one if the initial state is inactive (the first *if* clause) and at rate *ka20_p*;
d) the *send_degr_signal* process (made always available to the box by the replication process guarded by the input/output on the *rec2* channel) which is responsible for sending to other species in the system (not shown in the figure) degradation signal only if the state of the current box is active (the first *if* clause in the definition of the process);
e) the *activation_by_iep* process (made always available to the box by the replication process guarded by the input/output on the *rec3* channel) which is responsible for changing the states of the interfaces from the inactive to the active one if the initial state is inactive (the first *if* clause) and after an interaction on the channel *cdc20_act*, which is a communication channel used between CDC20 boxes and IEP ones (not shown in the figure);
Note that in the actual ".prog" file, the three code-extracts above should be written in the opposite order because the definition of the processes have to preceed its usage. However, for more clear reasoning and explanation, here they are listed the other way round.

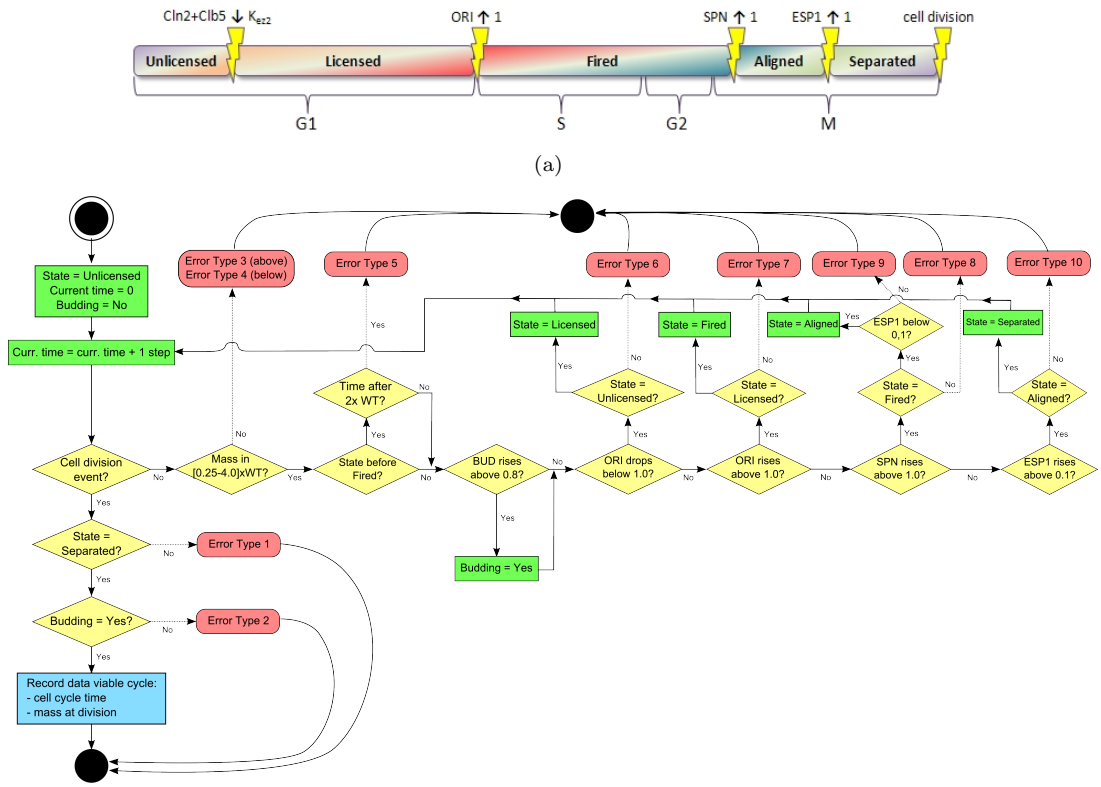
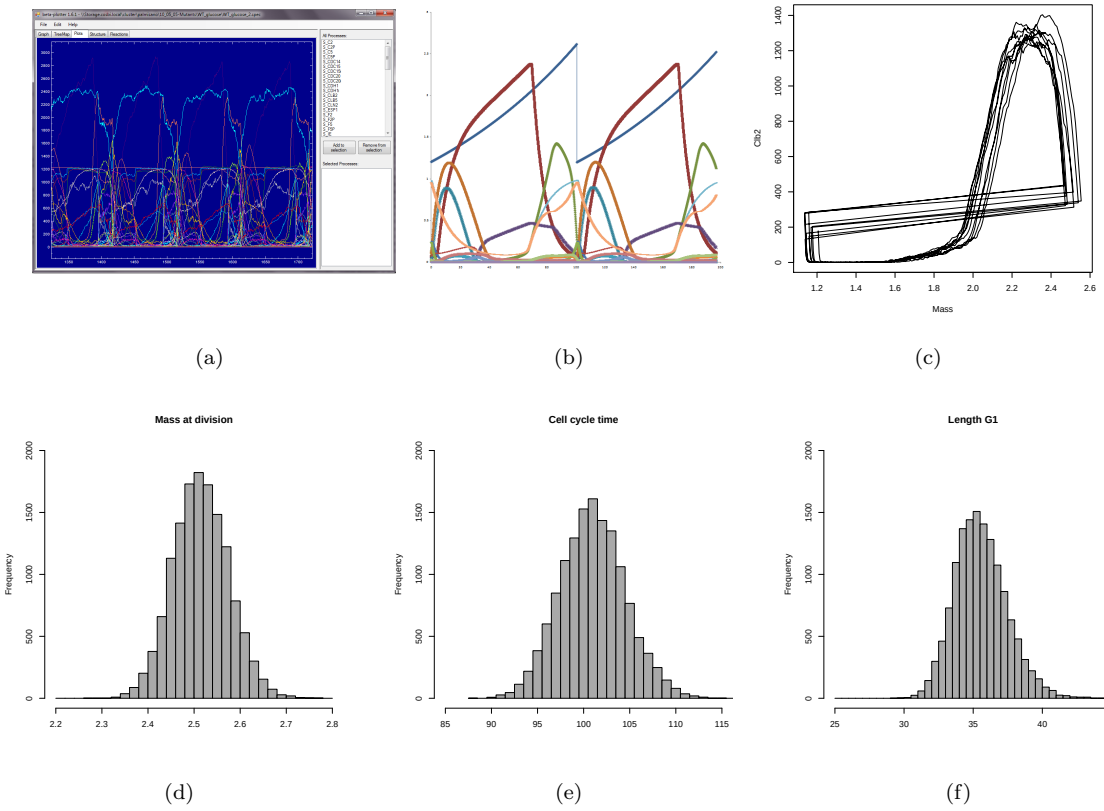


Figure 4.16: (4.16(a)) The five stages of the cell cycle, separated by the events described in the rules of viability of [146]. The up (down) arrow in the condition of the triggers (lightning) identifies a rise (drop) of the indicated quantity above (below) the threshold level on the right. The four biological phases of the cell cycle are indicated below the stages. (4.16(b)) Sequence of rules that determines the different arrest type of unviable cell cycles. The yellow diamond shapes represent condition checked at each time point, the green rectangles represent changes of state and the red rounded boxes record the error condition that classify the kind of unviability of the cycle currently under investigation.

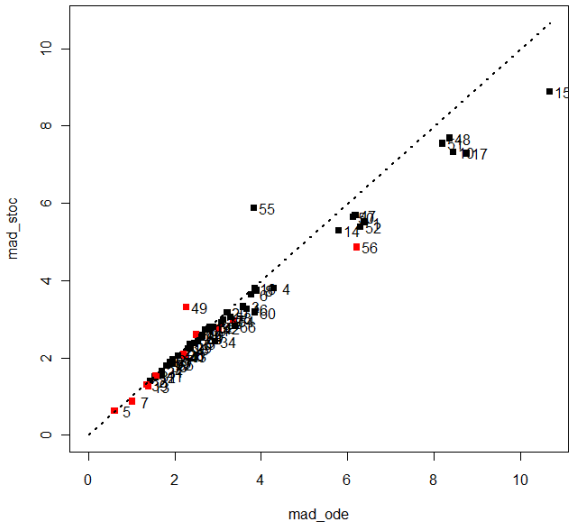
mutants are showed in Tables 4.1 (and summarized in Figure 4.18): obviously statistics about cell cycle time, mass at division, etc. are meaningful only for viable mutants, so for all the unviable ones we report (in Figure 4.19) the stage/reason of their death (always comparing the deterministic data with our stochastic simulation results; the deterministic data about all the mutants are reported in Appendix A.4 for reference).



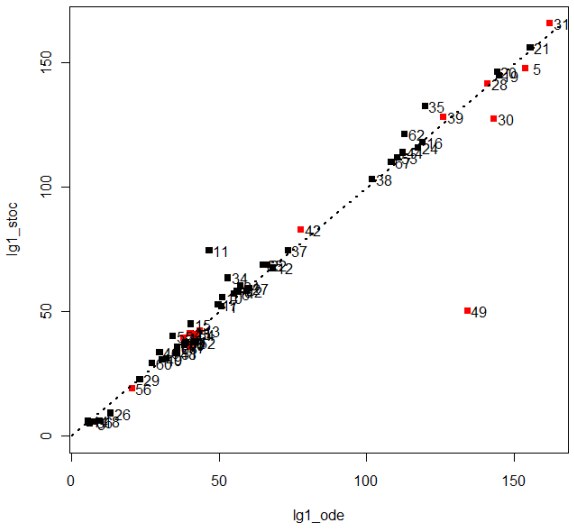
(a) (b) (c)

(d) (e) (f)

Figure 4.17: Statistics on different properties of the stochastic version of complete model of the budding yeast cell cycle. (4.17(a)) Visualization of the output of a BlenX simulation of the model in Appendix A.1. The time series plot is produced using the BetaPlotter tool, belonging to the CoSBiLab platform. The typical oscillatory behavior of the system can be easily seen together with the stochastic noise affecting the system. (4.17(b)) Solution of the original deterministic model. Only some species are reproduced just to show the classical oscillatory behavior of the whole system. (4.17(c)) Bifurcation diagram with some stochastic cell cycle trajectories. We plot the locus of Clb2 values (parametric in time) versus the increasing value of the mass in time. The abrupt jump from left to right indicates a cell division event. (4.17(d), 4.17(e), 4.17(f)) Statistics of mass at division, cell cycle time and length of G1 phase (respectively) on 100 runs of the stochastic model, following the daughter cell. Each simulation run reproduce about 150 complete cycles. The means of each collection of stochastic values are consistent with the results of the deterministic model (mean mad = 2.514964, mean cct = 101.2306, mean lg1 = 35.61656). Deviation from the mean values can be easily seen from the histograms and quantification of the spread of these distributions can be found in Table 4.1.



(a)



(b)

Figure 4.18: For all the viable mutants, a comparison of ODE values for mass at division (mad) and length of G1 (lg1) (taken from [19]) with the mean value of the same quantities calculated from the stochastic simulation runs of all the viable mutants (the numbers next to each point is referring to the "ind"ex of the specific mutant according to the list in Table 4.1). The general accordance between the two approaches can be seen in the overall trend of the value to be close to the 1:1 line. The red mutants are the ones for which the viability check module has been turned off (see text and Table 4.1 caption for discussions and further details).

ind	mutant name	mad	lG1	mad_{mean}	mad_{sd}	mad_{cv}	$lg1_{mean}$	$lg1_{sd}$	$lg1_{cv}$	ncycles
68	WT glucose	2.62	36.0	2.5150	0.0598	2.3797	35.6166	1.8671	5.2423	13800
67	WT galactose	1.92	108.4	1.8882	0.0345	1.8267	110.1978	4.1061	3.7261	6023
1	1 cln1D cln2D	6.39	51.0	5.5065	0.1221	2.2172	52.1977	2.2314	4.2748	13700
2	2 GAL CLN2 cln1D cln2D	1.43	59.4	1.3874	0.0419	3.0208	58.3421	2.9405	5.0402	605
3	3 cln1D cln2D sic1D	3.59	6.4	3.3215	0.0639	1.9241	5.0054	0.4918	9.8256	2689
4	4 cln1D cln2D cdh1D	4.29	56.7	3.7949	0.1173	3.0911	57.8980	2.6449	4.5683	13700
5	5 GAL CLN2 cln1D cln2D cdh1D	0.61	153.7	0.6192	0.0849	13.7054	147.7582	28.2126	19.0938	1396
6	6 cln3D	3.77	55.2	3.6352	0.0730	2.0084	57.2316	4.3262	7.5591	13700
7	7 GAL CLN3	1.02	40.3	0.8686	0.0685	7.8818	35.8271	2.5121	7.0117	5746
8	8 bck2D	3.9	56.2	3.7415	0.0879	2.3502	58.1555	4.9646	8.5368	13700
9	9 Multicopy BCK2	2.54	32.3	2.4186	0.0640	2.6464	30.9497	1.2150	3.9257	13721
10	10 cln1D cln2D bck2D	8.45	51.2	7.3175	0.1718	2.3473	55.7884	2.7106	4.8587	13600
11	13 cln3D bck2D GAL CLN2 cln1D cln2D	1.7	46.8	1.5415	0.0421	2.7325	74.6864	5.0757	6.7960	4934
12	16 cln1D cln2D cln3D GAL CLN2	1.55	68.4	1.5005	0.0414	2.7559	67.8229	3.6489	5.3801	2569
13	17 cln1D cln2D cln3D GAL CLN3	1.4	43.6	1.2587	0.0530	4.2145	42.1835	1.7768	4.2120	6900
14	18 cln1D cln2D cln3D sic1D	5.8	5.8	5.2972	0.1023	1.9312	5.9375	0.7932	13.3591	16

Table 4.1: (continued on the following pages) Stochastic characterization of all the viable mutants described in the deterministic framework. The first column is an index used in Figure 4.18 for referring to the specific mutant. The second column contains the mutant name (using name conventions similar to the ones on the web-site [19]). The third and fourth columns contain the mass at division (mad) and length of G1 (lg1) values of the deterministic model (taken from [19]). The following columns contain their stochastic counterpart (with mean, standard deviation and coefficient of variation). The last column records the number of viable cycles collected running a batch of 100 simulations of the model. Before collecting the values for the different cell cycle properties, every cycle has been analyzed by a "viability check" module, implementing the rules in Figure 4.16, and only the cycles that was marked as *viable* have been used for producing the statistics in the table above. Because of the fact, that all the cycles of the wild type model (in glucose or galactose media) resulted to be viable, their total number of cycles should be considered as a reference number for evaluating the survival capability of all the other mutants (in the same growth media). Small deviation from the "ncycles" number denote a mutant with good survival rate (the small difference can just be caused by the small random variation of the cell cycle time in the fixed length of simulation). Bigger deviation is indicative of partially viable mutant which can be an interesting subject for deep study in the stochastic framework (because they can be more sensible to the effect of the noise with respect to the wild type case).
The underlined mutants are peculiar mutants where the "viability check" module had to be disabled in order to have more than zero cycles to collect statistics: since, in this way, some of them seem to get back the consistency with respect to their deterministic values, this suggests that some viability rules and thresholds need to be relaxed/changed to account for the stochastic variation introduced by the noise. Moreover those mutants are interesting subjects for further analysis in the stochastic framework (see Section 7 for further discussions and ideas).

ind	mutant name	mad	lG1	mad_{mean}	mad_{sd}	mad_{cv}	lg1_{mean}	lg1_{sd}	lg1_{cv}	ncycles
15	20 cln1D cln2D cln3D multi copy CLB5	10.68	40.6	8.8873	0.2188	2.4623	45.0108	2.2209	4.9343	13600
16	21 cln1D cln2D cln3D GAL CLB5	3.86	119.0	3.7898	0.0794	2.0956	118.1869	5.5344	4.6827	6800
17	22 cln1D cln2D cln3D multicopy BCK2	8.75	49.8	7.2874	0.1831	2.5128	52.8218	2.5553	4.8376	13600
18	25 sic1D	2.32	9.7	2.2283	0.0564	2.5324	6.0293	0.7004	11.6170	13800
19	26 GAL SIC1	2.81	144.8	2.7849	0.0364	1.3057	144.9432	3.8122	2.6301	616
20	29 GAL SIC1 GAL CLN2 cln1D cln2D	3.22	144.2	3.1654	0.0426	1.3443	146.3507	4.2073	2.8748	6800
21	31 GAL SIC1 GAL CLN2 cln1D cln2D cdh1D	2.8	155.4	2.7350	0.0509	1.8616	156.2028	6.1067	3.9095	6857
22	32 cdh1D	1.91	66.3	1.8419	0.0278	1.5120	68.8062	2.1425	3.1138	13800
23	34 sic1D cdh1D	0.0	0.0	2.7863	0.2962	10.6301	12.4369	3.1788	25.5591	2447
24	35 sic1D cdh1D GALL CDC20	1.7	117.5	1.6562	0.0461	2.7862	115.8675	27.4266	23.6706	381
25	36 cdc6D2 49	2.46	40.6	2.3696	0.0554	2.3385	41.3001	2.6172	6.3369	13800
26	37 sic1D cdc6D2 49	2.18	13.4	2.0796	0.0648	3.1172	9.1549	1.0346	11.3006	12959
27	38 cdh1D cdc6D2 49	1.96	60.2	1.9515	0.0643	3.2954	59.3719	3.8502	6.4848	13800
28	40 sic1D cdc6D2 49 cdh1D GALL CDC20	2.2	141.0	2.0886	0.1214	5.8104	141.6004	17.9365	12.6670	6088
29	41 swi5D	2.37	23.3	2.2832	0.0619	2.7097	22.8780	4.0202	17.5723	12078
30	44 swi5D cdh1D GAL SIC1	2.5	143.0	2.5973	0.3367	12.9644	127.4746	34.2982	26.9059	413
31	47 GAL CLB2	1.57	162.0	1.5379	0.0477	3.0991	165.9196	8.9384	5.3872	6900
32	49 CLB1 clb2D cdh1D	0.0	0.0	1.7197	0.0245	1.4257	74.0836	2.1235	2.8664	13800
33	50 CLB1 clb2D pds1D	0.0	0.0	3.8303	1.1150	29.1097	22.8451	25.3729	111.0649	1284
34	55 CLB2 dbD multicopy SIC1	2.94	53.0	2.4234	0.0514	2.1214	63.5516	2.4673	3.8824	13800
35	56 CLB2 dbD GAL SIC1	3.08	119.8	2.9018	0.0823	2.8368	132.7047	7.3738	5.5566	6800
36	57 CLB2 dbD multicopy CDC6	0.0	0.0	3.5344	0.3928	11.1147	34.1533	1.8214	5.3329	75
37	61 cllb5D clb6D	3.13	73.4	2.9773	0.0808	2.7153	74.7180	3.5575	4.7613	1927
38	63 GAL CLB5	1.82	101.9	1.7933	0.0350	1.9501	103.1897	3.9261	3.8047	6900
39	65 GAL CLB5 cdh1D	1.36	126.0	1.3043	0.0344	2.6380	128.2228	7.0532	5.5007	6900
40	66 CLB5 dbD	2.64	30.8	2.5745	0.0653	2.5364	30.6268	1.9476	6.3591	13800

Table 4.1: (continued)

ind	mutant name	mad	lG1	mad_{mean}	mad_{sd}	mad_{cv}	lg1_{mean}	lg1_{sd}	lg1_{cv}	ncycles
41	68 CLB5 dbD pds1D	2.24	38.1	2.0464	0.0988	4.8291	39.3948	13.4373	34.1092	13800
42	74 cdc20D pds1D cllb5D	3.02	77.8	2.7612	0.0891	3.2266	82.8696	4.6569	5.6195	13800
43	78 pds1D	2.25	40.2	2.0417	0.0927	4.5385	41.1202	11.7610	28.6015	13800
44	85 GAL TEM1	2.08	112.2	2.0429	0.0394	1.9304	113.9415	4.9057	4.3055	710
45	86 tem1ts multicopy CDC15	0.0	0.0	3.7497	0.2551	6.8031	24.8879	1.3746	5.5230	13700
46	87 tem1ts GAL CDC15	3.66	30.0	3.2582	0.3506	10.7597	33.5230	5.9575	17.7714	6804
47	88 tem1D net1ts	6.18	38.5	5.6884	0.1173	2.0627	36.7148	1.8840	5.1314	13700
48	89 tem1D multicopy CDC14	8.36	35.8	7.6817	0.2114	2.7513	33.1939	2.2273	6.7100	13600
49	91 Multicopy CDC15	2.27	134.1	3.3156	0.1031	3.1096	50.3903	2.6892	5.3367	13701
50	93 cdc15D net1ts	6.13	38.5	5.6479	0.1131	2.0029	36.7946	1.8613	5.0586	13700
51	94 cdc15ts multicopy CDC14	8.2	36.0	7.5438	0.2117	2.8068	33.3358	2.2199	6.6593	13600
52	95 net1ts	6.3	42.1	5.3926	0.1207	2.2380	37.7778	2.3097	6.1140	36
53	99 GAL NET1 GAL CDC14	1.9	110.5	1.8772	0.0344	1.8301	111.8251	4.1441	3.7059	1618
54	106 TAB6 1	3.35	42.0	2.9497	0.0785	2.6606	40.7095	1.9553	4.8030	13780
55	107 TAB6 1 cdc15D	3.83	34.5	5.8737	0.0429	0.7309	40.1667	3.8188	9.5074	3
56	109 TAB6 1 CLB1 clb2D	6.22	20.6	4.8609	1.1016	22.6629	19.2313	20.5566	106.8912	10748
57	110 mad2D	2.35	38.9	2.3398	0.0674	2.8796	37.7057	3.1052	8.2353	12518
58	111 bub2D	3.29	38.6	3.0512	0.0929	3.0459	37.2882	1.9370	5.1947	13732
59	112 mad2D bub2D	2.7	38.8	2.7207	0.0942	3.4635	37.4504	2.0441	5.4582	13669
60	122 APC A	3.86	27.4	3.1775	0.1186	3.7319	29.3006	1.1763	4.0147	12185
61	125 APC A cdh1D multicopy SIC1	0.0	0.0	2.9116	0.1047	3.5965	56.3892	2.4893	4.4144	13782
62	126 APC A cdh1D GAL SIC1	2.89	112.8	2.7898	0.1065	3.8183	121.3874	10.3666	8.5401	6717
63	127 APC A cdh1D multicopy CDC6	0.0	0.0	2.6943	0.0445	1.6533	31.8232	1.3441	4.2238	1055
64	128 APC A cdh1D GAL CDC6	2.63	57.2	2.5411	0.1087	4.2773	60.2407	6.0149	9.9848	6831
65	129 APC A cdh1D multicopy CDC20	1.96	65.1	1.8403	0.0339	1.8444	68.8798	3.9287	5.7037	13800
66	130 APC A sic1D	3.39	7.4	2.8185	0.1059	3.7574	5.8275	0.5794	9.9424	13797

Table 4.1: (continued)

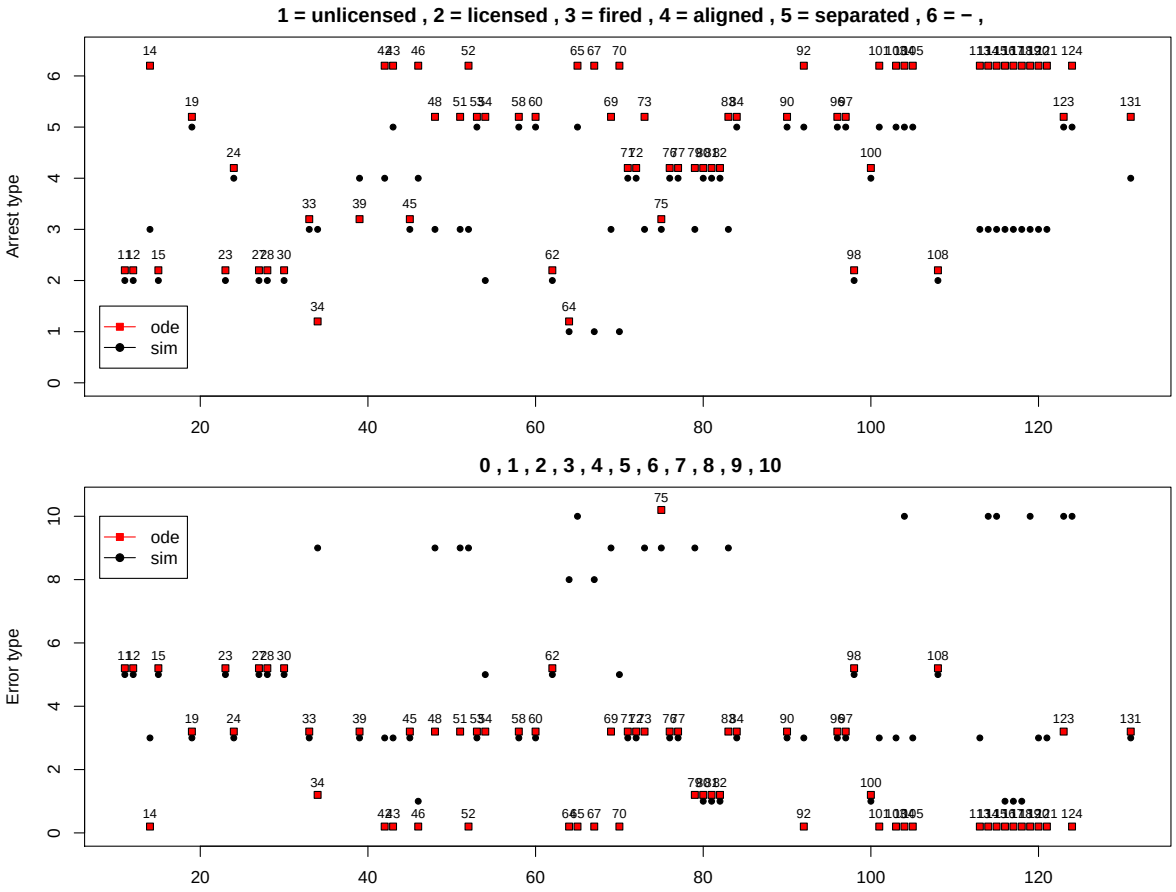


Figure 4.19: For all the inviable mutants, a comparison of the arrest/error type deduced from the deterministic solution of the model (see [146]) and stochastic runs of the same parameter sets. On the x-axis a numeric identifier is associated with each mutant (see Appendix A.4 for associating the number with the mutant they refer to). There are some discrepancies between the deterministic classification of the inviable mutants and the stochastic ones. Most of them are due to the fact that in the deterministic case, no justification of the cause/stage of death is recorded in [146]. Other inconsistencies between the results of the two approaches can be ascribed to the sensitivity of some peculiar mutants to the threshold of the viability check module. See text for details.

As we can see in Table 4.1 there are some mutants whose stochastic behavior nicely resemble that of their deterministic counterpart, while some others show some peculiar statistics. We can identify two kind of inconsistencies:

1. (with respect to the corresponding wild type parameter set) some mutants show a lower number of viable cycles on which the statistics has been taken;

2. (with respect to the deterministic solution) some mutant seems to be so delicate that the viability rules used in the general case consider unviable all the simulated cycles so, in order to have some statistics, the viability check module has been turned off

for those mutants (the underlined values in the tables above).

Both type of inconsistencies are due to the application of the viability check rules (Figure 4.16(b)) on the collected simulated cycles: however, if the difference in the number of collected cycles between a specific mutant and its corresponding wild type parameter set is in the order of a few hundreds, this is just the consequence of the fact that a slightly greater (smaller) cell cycle time can result in a lower (higher) total number of cycles (in a fixed simulation time). If the difference in the number of viable cycles, instead, is more than a few hundreds, this implies that the considered mutant is not completely viable and this is a signal that the mutant needs a more careful study because the effect of the stochastic noise is not negligible (so, for example, a quantification of its unviability can be done on its simulation results). As an extreme case of this situation, stands the second inconsistency described above: if the number of "supposed to be" viable cycle is zero this means that the mutant is quite peculiar and/or it needs different rules for defining its viability (see Figure 4.20 for an example where the threshold level should be relaxed in order to take into consideration the noise affecting the system).

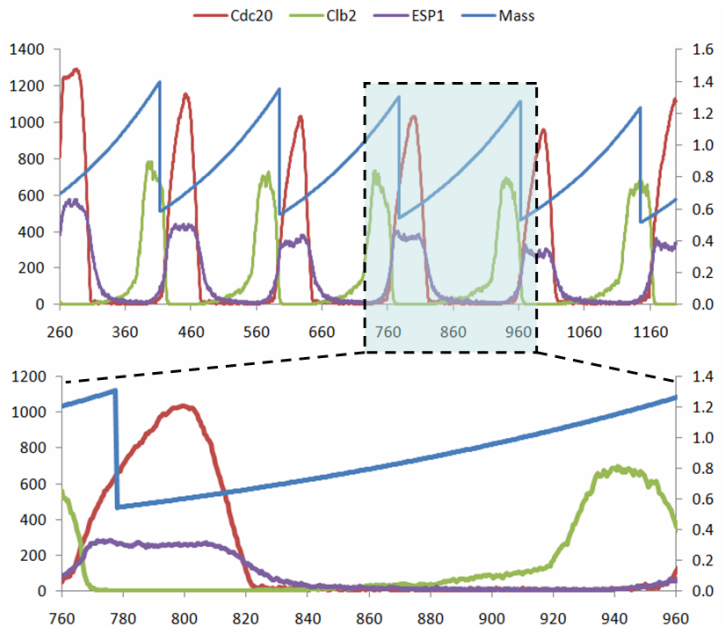


Figure 4.20: A simulation of *cln1D cln2D cln3D GAL CLN3* mutant (Mass and Esp1 axis on the right, expressed in concentration; Cdc20 and Clb2 axis on the left, expressed in number of molecules). This mutant is one of the peculiar ones in Table 4.1 for which the viability check has to been disabled in order to be able to obtain meaningful statistics. As we can see from the top plot, the mutant seems to be oscillating and dividing; however the viability rule of "ESP1 rises above 0.1 before division" has not been fulfilled in the cycle depicted in the bottom row, so the division happens when the stage of the cycle is still "aligned" and not yet separated, so this cycle (and all the following ones) should be considered inviable.

A careful study of all those mutants is an interesting future development of this work, because these results confirm the importance of considering the effect of the noise on this system when mutants at the border of life and death are analyzed. As we will see in the next section, there is a further interesting study that can be done through the stochastic framework and that can reveal characteristics of some mutants that cannot be seen by the deterministic framework.

4.5 Stochastic pedigree analysis of Chen model

As we saw in the previous section, analysing a stochastic model of the budding yeast cell cycle can reveal interesting properties of the system that are more in accordance with experimental measures with respect to the conclusions that can be drawn from deterministic studies. However the statistics presented in the previous section are collected from batch of single simulation runs, starting from the same initial conditions: this means that each cycle in a run is correlated to the previous one because its initial state is uniquely determined by the state of the previous cycle in the moment when the event of division happens. So putting together all the single cycles from all the simulation runs and considering them as independent sample cycles from a population is not exactly correct. However following one initial cell long enough in its continuous growth/division can resemble the population trend (because, due to the robustness of the biological process, the effect of the initial conditions are lost after some transient cycles).

In order to do proper statistics on the system under consideration and because of the fact that recent advances in experimental techniques allow us to have data about single cell measurements [28, 29, 40], we decided to implement a framework in which it is possible to follow the development of a single cell and its own offspring. So what we are going to present in this section is a framework that we called $\mathsf{spPeAn}^{\mathrm{BY}}$: Stochastic Parallel Pedigree Analysis (for Budding Yeast).

The division of a budding yeast is asymmetrical, giving two cells of different sizes [125]: this happens because at the "Start" transition of the cell cycle, a bud emerges from the mother cell, and subsequent cytoplasmic growth is directed primarily to the bud. S and M phases of the cycle are completed before the bud grows as large as its progenitor; thus cell separation produces a large mother cell and a small daughter cell.

The basic idea behind $\mathsf{spPeAn}^{\mathrm{BY}}$ is to be able to run a simulation of a single cell cycle (from an initial model), then – after the first division event – to apply different division rules for having mother and daughter cell (models), to run (possibly in parallel) the simulations for both models and, at their first division, start again the process (up

to a fixed number of generations). This allows the user to end up with a complete and detailed characterization of the growth of a specific initial cell, in the same way as it is happening when, in a wet-lab, the experimentalist is following the growth of a colony of cells starting from a single cell (see Figure 4.21). Indeed when yeast cells bud, a ring of chitin builds up at the bud isthmus. This chitin ring remains on the parent cell after the bud (new born daghter cell) has separated from the parent, and it is then termed "bud scar". Bud scars can be visualized by fluorescence microscopy: it is therefore possible to follow each cell in its growth and division history and to ascertain its age in term of how many cycles it has passed through from the number of bud scars it possesses.

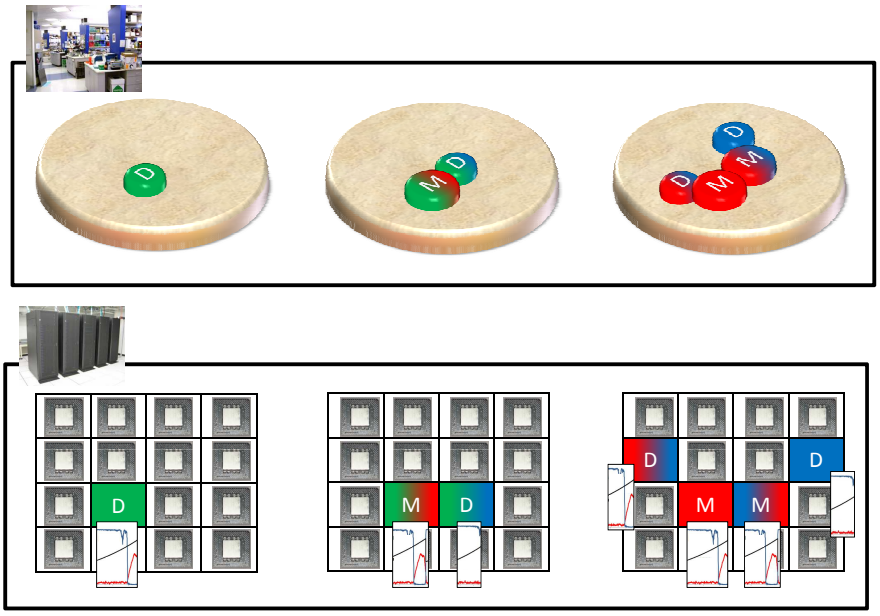


Figure 4.21: A cell growing and dividing for three generations on a Petri dish (top) and, in parallel, on different cores of a cluster of computer (bottom). The color codifies the link of each cell to its parents: the initial green cells duplicates and generates a mother (green-red) and a daughter (green-blue) cells; the following generation uses similar color-patterns.

4.5.1 spPeAn^{BY} architecture

The workflow that spPeAn^{BY} implements to simulate in silico the study described above is depicted in Figure 4.22: the initial input model files (in our work the three BlenX files, .prog, .types and .func) are put in a queue that is continuously inspected by the "Cluster manager" (C). It takes these files and uses them to start the BlenX simulations, invoking the "BetaWB Simulator" (S) on them. As soon as the simulations are finished, the cluster

manager puts the output of the simulations in another queue that is continuously inspected by the "Generator of offspring" module (G). This module is performing a different kind of analysis on the simulation outputs and, according to the result of some checks, it generates the model files for the next generations. In our case, the checks are of two kinds: is the just finished cycle viable (according to the rules listed in section 4.4)? if the answer is yes, then its ending conditions are used, by the "Division" module (D), to produce (according to some rules explained in the next section) the new .func files containing the parameters (and initial conditions) of the mother and daughter cell model: the old .prog and .types are put again in the input queue together with the new .func files, so that the cluster manager can take them and restart the whole process. If the cycle is not viable, its offspring is not added to the main input queue, and this will result in a terminated branch of the pedigree tree that the whole procedure is producing (see Figures 4.23). The other terminating condition of the G module happens when the maximum number of generations requested by the user is reached.

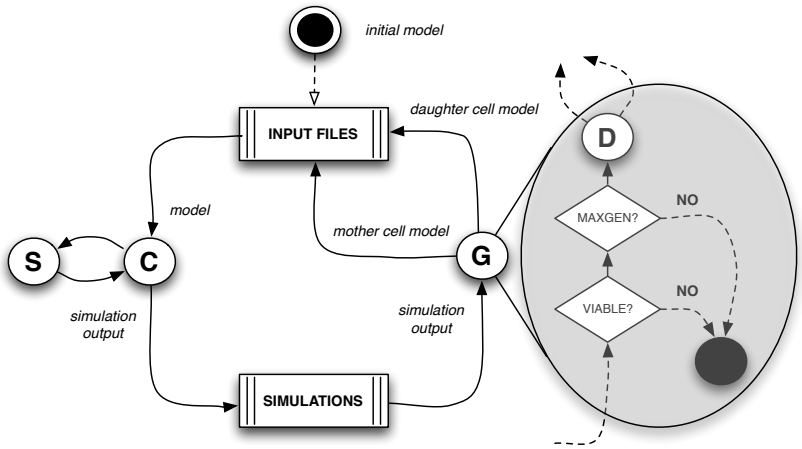


Figure 4.22: spPeAn^{BY}architecture. See text for details.

Before going ahead showing the results obtained by spPeAn^{BY}on the chosen case study, we want to point out that the implemented framework is pretty general and it can be applied to any system whose main characteristic is to start as a single entity and, after a while, for some reasons, with some specific rules, generates (possibly many) indipendent offspring. In this work it has been specialized on stochastic simulations of the Chen model of budding yeast implemented in BlenX but, thanks to the architecture of the whole project, it can be easily adapted to other biological systems.

4.5.2 spPeAn^{BY}in action

One of the main reasons behind the need for a detailed pedigree analysis of budding yeast cell cycle is its primary characteristic of having an unequal division of the content of the cell, when the separation of the mother and the daughter cells occurs. This asymmetrical divison has been experimentally verified since the early 80s by Lord and Wheals [125], who tested this feature by bud scar analyses of cultures of *Saccharomyces cerevisiae* growing at constant temperature at 19 different rates (obtained by altering the carbon source). With these analyses the authors conclude that daughter cycle time, parent cycle time, and the budded phase are all linearly related to the growth rate and they converge to equality (symmetrical division) at growth rate equal to 65 min.

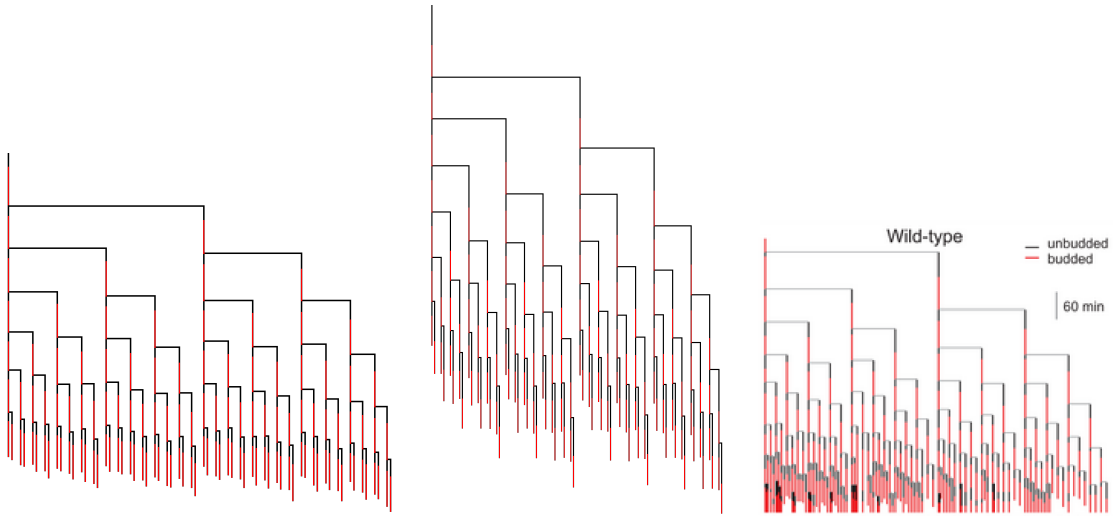
The physiology behind the asymmetrical division (as explained in [31]) is the following: when a cell cycle starts, a bud emerges from the mother cell so the cytoplasmic growth is directed primarily to the bud. S and M phases of the current cell are completed before the bud grows as large as its progenitor: so cell separation produces a large mother cell and a small daughter cell. Shortly after division, the mother cell produces a new bud, but the daughter cell enters an extended G1 phase, during which it has to grow to a critical size before being able to produce a bud of its own. As the experiments by Lord and colleagues show, the whole process is quite sensitive to growth rate: at the fastest growth rates, division is almost symmetrical, as growth rate is decreased, cell division becomes increasingly asymmetrical.

This informal description of the process has been encoded in the Chen model ([30]) with the following rule for calculating the new mass at division: defining f as the fraction of mass given to the daughter at cell division, they choose f to give the observed daughter cycle time (D) at any particular growth rate ($\mu = 0.693/T_d$ where T_d is the mass doubling time). From the assumption that cells grow exponentially, (mother size at division) = (daughter size at birth)$\cdot e^{\mu \cdot D}$, so f = (daughter size at birth)/(mother size at division) = $e^{\mu \cdot D}$. By using the empirical formula for daughter cycle time, $D = 1.48 \cdot T_d - 32$ ([125], Table 2) to calculate f, the model fits the data for D as a function of T_d. With the above values of f, it is enough to set (new mass) = $f\cdot$ (mass at cell separation) in order to follow the behavior of the daughter cell, and the (new mass) = $(1-f)\cdot$ (mass at cell separation) in order to follow the behavior of the mother cell.

In our BlenX model, for the analysis presented in the previous section, we used this same approach, following the simulations of the daughter cell. However, in the pedigree analysis we are allowed to follow both mother and daughter cells and, moreover, we applied the f function not only to the value of the mass, but also to the level of all the proteins present at the moment of division in the cell: in this way we are accurately modelling the split of the whole content of the cell.

In the first version of the spPeAn^{BY} (used to produce the plots in Figure 4.23 and the following statistics in Figure 4.26-4.27) the f function is applied in a deterministic way. However, the framework of spPeAn^{BY} is easily extendable to allow the addition of different kinds of extrinsic noise on any of the following parts of the model: species concentration level, initial mass, growth rate, other parameters – in a dependent or independent way – see Chapter 7 for some discussions about those future developement.

In the rest of this section, we will show some interesting analyses that have been performed on the output generated by spPeAn^{BY}on the Chen model.



(a) (b) (c)

Figure 4.23: Pedigree analysis of the stochastic version of the Chen model, with parameter set as wild type in glucose (4.23(a)) and wild type in galactose (4.23(b)). The plots generated from the simulations are comparable with the experimental phylogenies (4.23(c)) showing the budded/unbudded periods for all descendents of a founder cell as the ones in [28].

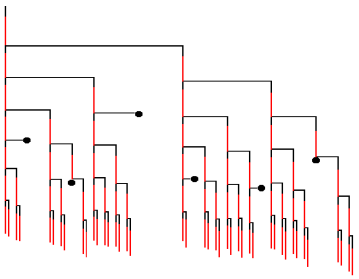


Figure 4.24: An example of an incomplete pedigree tree. The black dots indicate that the simulation of that cycle was considered inviable and, due to that, all its offspring have been neither generated nor simulated. When spPeAn^{BY} completes the simulations of the desired number of generations for a specific mutant, statistics about the probability of survival (i.e. the number of healthy cycle versus the dead ones) can be computed and compared with similar experimental measures.

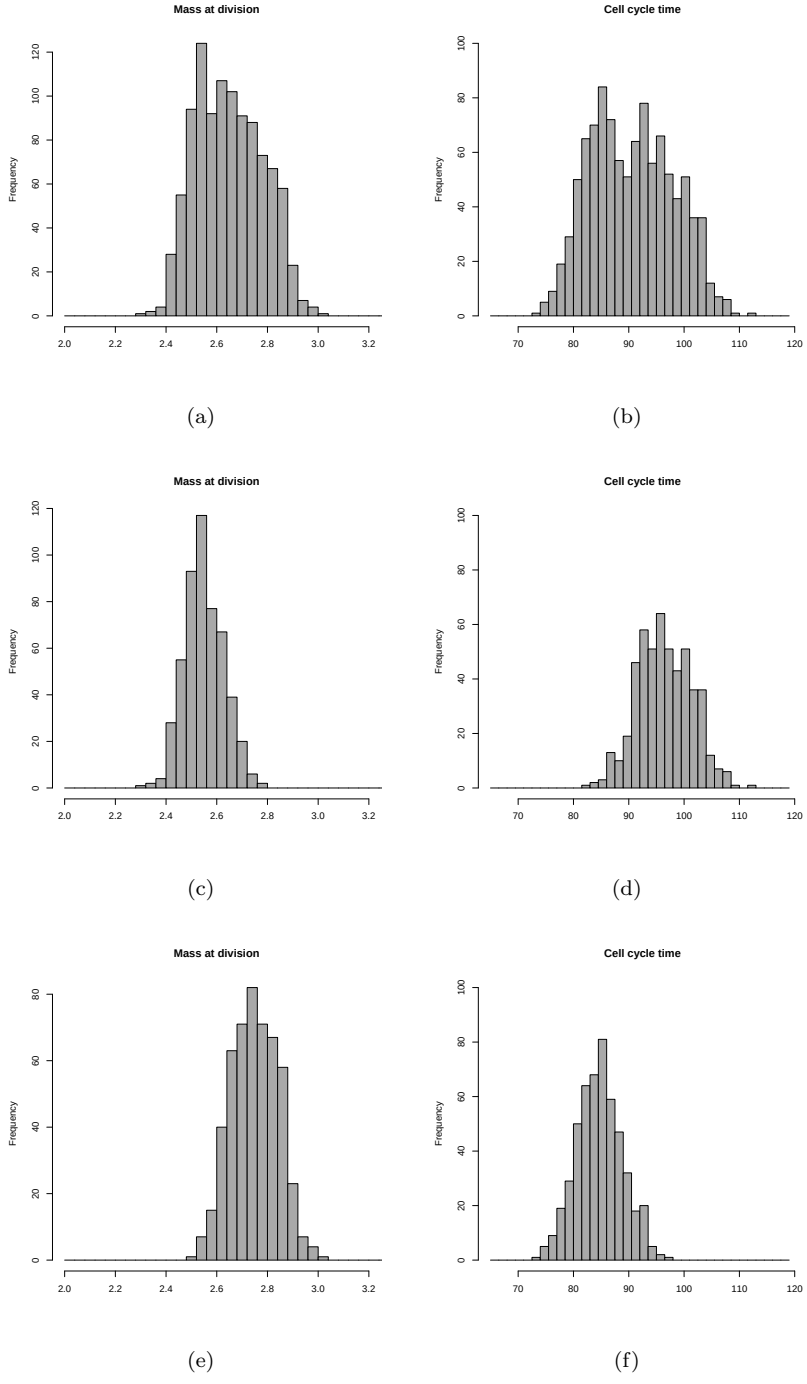


Figure 4.25: Statistics on the output of the pedigree analysis of the stochastic version of the Chen model. Mass at division and cell cycle time statistics are shown collecting data from all the available cycles (top row), just for daughter cells (central row) and just for the mother cells (bottom row): the asymmetrical behavior of the two branches of the pedigree tree can be easily seen in the double gaussian distribution of cell cycle time, while it is less evident (even if present) in the statistics about the mass.

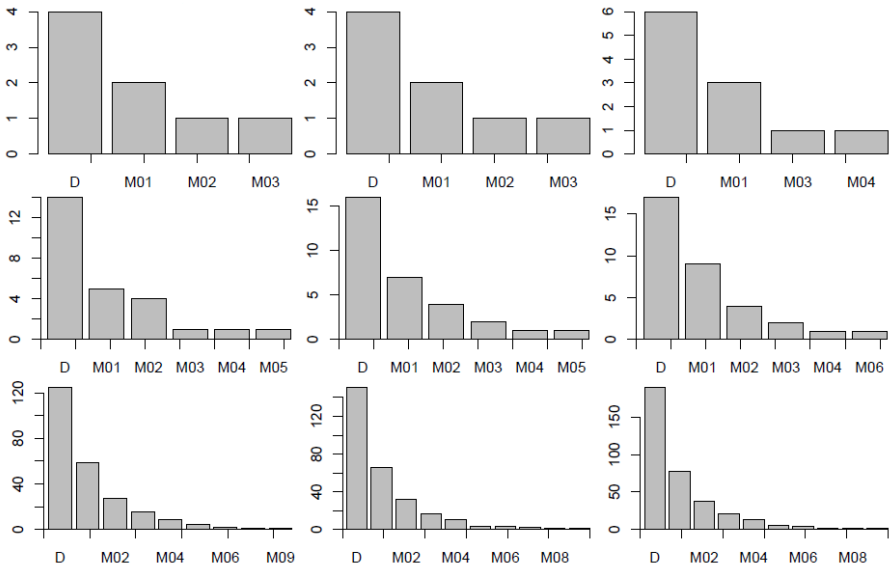


Figure 4.26: Histograms of the genealogical classification of a colony of budding yeast cells growing in glucose media. The data have been obtained cutting the pedigree tree in Figure 4.23(a) at random consecutive time instants. The expected distribution of daugthers (D) and mother (M) of generation 01, ..., 09 has been found.

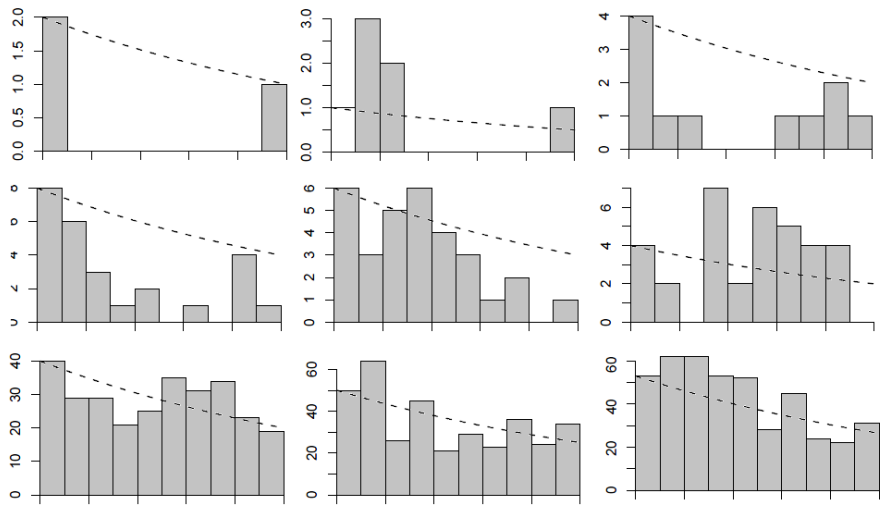


Figure 4.27: Histograms of the synchronicity of a colony of budding yeast cells growing in glucose media. The data have been obtained cutting the pedigree tree in Figure 4.23(a) at random consecutive time instants and measuring the percentage of advance in the cell cycle (where 0 and 100 represent the division event). The asynchronicity of the colony can be seen by the fact that the distribution of the frequency of the number of cells at the different stages is approaching the ideal age distribution (the dashed line in each plot).

4.6 Some concluding considerations

Let us conclude this part of the thesis summarizing the analysis and the results that have been carried out on the specific case study of budding yeast cell cycle.

The first result that we were able to obtain is a quantification of the probability that the "Start" transition of the cell cycle is irreversible, a property that has never been tested before. We showed that by weakening the strength of the positive feedback loop (by reducing a specific parameter of the model) the irreversibility gets lost. With stochastic simulations of the budding yeast cell cycle model we showed that indeed the above mentioned parameter variations can perturb the irreversibility of the "Start" transition of the cell cycle.

The second result is obtained through the application of the semi-automatic procedure, explained in Section 3.2, that allows us to translate an existing model written with ODE into BlenX. The whole process have been applied to a budding yeast cell cycle model and we were able to show that, in the wild type case, the stochastic simulations are consistent with the results obtained by the solution of the ODE system using the same parameter set, but the stochastic model is also able to explain peculiar behaviour of some mutants that present viability issues. We chose to analyse a nutritionally sensitive mutant (Clb2dbΔ clb5Δ) for which experimental results show that it is viable only with a decreased growth rate. Both the deterministic and the stochastic model results correctly show its viability at a low growth rate, but it is interesting to observe that the deterministic model cannot predict the intermediate situations between a high growth rate (modelling a growth medium which is lethal for the mutant) and another growth rate modelling a medium which allows it to be completely viable. It is instead reasonable to expect a continuous transition as the growth rate varies in the interval, with some mutant cells that have a limited survivability for values of the growth rate inside the interval and a death probability increasing approaching the lethal situation. This was indeed the result that we obtained studying the stochastic simulations of the BlenX model of this mutant.

The third result presented in the previous chapter is a full characterization of the most complete model of the budding yeast cell cycle available [30]. With this analysis we wanted to acquire, in the stochastic framework, the detailed knowledge about this system that has been made available through the study in the deterministic framework. In particular we characterized the set of 131 mutants that can be found in [19]. Some of those mutants are viable, while some others are not, so beyond the standard stochastic simulation runs, we implemented a program that analyzes the stochastic traces produced by the BetaWB simulator and it is able to tell, according to the rules in [146, 2], if a cycle has to be considered alive or not and (if not) what is the reason/stage of its death. This

kind of analysis allow us to identify some delicate mutants that may need more careful studies both with in-silico and in-vitro experiments.

The final result that we obtained was motivated by the consideration that even if analysing a stochastic simulations of a model can reveal interesting properties of the system that are more in accordance with experimental measures with respect to the conclusions that can be drawn from deterministic studies, the statistics collected from batch of single simulation runs, starting from the same initial conditions can be inadequate. In particular this consideration it is due to the fact that each cycle in a run is correlated to the previous one because its initial state is univocally determined by the state of the previous cycle. So putting together all the single cycles from all the simulation runs and considering them as independent sample cycles from a population is not exactly correct. So in order to do proper statistics on the system under consideration we implemented a framework in which it is possible to follow the development of a single cell and its own offspring. Using this framework we were able to create pedigree trees of simulation runs where each branch represents the stochastic trace of a single cycle. Analysis of these pedigree trees resemble the kind of results that recent advances in experimental techniques allow us to obtain about a single cell of budding yeast.

Part II

The inference framework

Chapter 5

KInfer: a new tool for parameter estimation of biochemical systems

The construction of a model consists of two tasks: 1. deciding on the model structure and 2. estimating the involved parameters values. The previous chapters detailed some approaches for the former issue, while the following chapters are focused on the key step of parameter estimation, assuming the structure of the model as given.

Parameter estimation (also known as model calibration) from experimental data is a bottleneck for a major breakthrough in computational systems biology because inter- and intra-cellular processes require dynamic models, that contain the rate constants of the biochemical reactions. However the kinetic rate constants governing the dynamical behavior of the system in time are frequently not accessible directly through experiments, therefore methods that estimate rate constants with the maximum precision and accuracy are needed. Parameter inference aims to find the parameters of the model which give the best fit to a set of experimental data. In this chapter we present a method for parameter estimation based on a probabilistic, generative model of the variations in reactant concentrations.

The overall workflow of the proposed strategy can be summarized as follows: we take the observed time series of concentrations for all the reactant species defining the model under consideration and we gather them in N state vectors $\mathbf{X}_1, \ldots, \mathbf{X}_N$. Then, the state vectors and the structure of the model written following the general mass action formalism are used to define a probability density function of the observed increments/decrements for all the species in the system: the function is expressed in terms of the unknown kinetic constants so this likelihood for the observed increments/decrements can be optimized with respect to the rate coefficients of the biochemical network under consideration.

5.1 The model for inference

Consider N reactant species, $S_1, S_2, \ldots, S_N$, with concentrations $X_1, X_2, \ldots, X_N$, that evolve according to a system of rate equations

$$\frac{dX_i}{dt} = f_i(\mathbf{X}^{(i)}(t); \theta_i) \tag{5.1}$$

where θ_i $(i = 1, 2, \ldots, N)$ is the vector of the rate coefficients, which are present in the expression of the function f_i. $\mathbf{X}^{(i)}$ is the vector of concentrations of chemicals that are present in the expression of the function f_i for the species i. We wish to estimate the set of parameters $\mathbf{\Theta} = \cup\theta_i$ $(i = 1, 2, \ldots, N)$, whose element θ_i is the set of rate coefficients appearing in the rate equations of i-th species (therefore $\theta_1 = \{\theta_{11}, \theta_{12}, \ldots, \theta_{1N_1}\}, \ldots, \theta_N = \{\theta_{N1}, \theta_{N2}, \ldots, \theta_{NN_N}\}$).

According to the law of mass action, the functions f_i have the general form:

$$f_i(\mathbf{X}^{(i)}(t); \theta_i) = \theta_{i1} \prod_{w \in S_1} X_w^{\alpha_w} + \cdots + \theta_{iN_i} \prod_{w \in S_{N_i}} X_w^{\alpha_w} = \sum_{h=1}^{N_i} \left(\theta_{ih} \prod_{w \in S_h} X_w^{\alpha_w} \right)$$

where $\alpha_w \in \mathbf{R}$, and N_i is the number of parameter in the f_i rate equation. The rate equations in (5.1) form the so-called Generalized Mass Action law [126].

We assume that we have a number M of concentration measurements for each considered species. We also assume that we have noisy observations $\hat{X}_i = X_i + \epsilon$ at times $t_0, \ldots, t_M$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian noise term with mean zero and variance σ. With this choice we are assuming that the concentration measurements are not significantly affected by systematic errors, but by uncontrolled random errors and that an error is equally likely to occur in either positive or negative direction with respect to the symmetry axis of the distribution.

Approximating the rate equation (5.1) as a finite difference equation between the observation times, we have:

$$X_i(t_k) = X_i(t_{k-1}) + (t_k - t_{k-1}) \cdot f_i(\mathbf{X}^{(i)}(t_{k-1}); \theta_i) \tag{5.2}$$

where $k = 1, \ldots, M$. In Eq. (5.2) the rate equation is viewed as a model of increments/decrements of reactant concentrations; i.e., given a value of the variables at time t_{k-1}, the model can be used to predict the value at the next time point t_k. Increments/decrements between different time points are conditionally independent by the Markov nature of the model (5.2). Therefore, given the Gaussian model for the noise, it is possible to estimate the probability to observe the value $\hat{X}_i(t_k)$ given the model at time

t_{k-1} ($X_i(t_{k-1})$) and the set of parameters θ_i, as:

$$p\Big(\hat{X}_i(t_{k-1})|X_i(t_{k-1})\Big) = \mathcal{N}\Big(X_i(t_{k-1}) + (t_k - t_{k-1}) \cdot f_i(X_i(t_{k-1}, \theta_i), \sigma^2\Big) \tag{5.3}$$

We then also have that the true value of $X_i(t_k)$ is normally distributed around the observed value $\hat{X}_i(t_k)$, so that

$$p\Big(X_i(t_{k-1})|\hat{X}_i(t_{k-1}))\Big) = \mathcal{N}\Big(\hat{X}_i(t_{k-1}), \sigma^2\Big) = \frac{1}{\sqrt{2\pi}\sigma} \exp\Big[-\frac{(X_i(t_{k-1}) - \hat{X}_i(t_{k-1}))^2}{2\sigma^2}\Big] \tag{5.4}$$

Therefore, the probability to observe a variation $D_i(t_k) = X_i(t_k) - X_i(t_{k-1})$ for the concentration of the i-th species between the time t_{k-1} and t_k, given the parameter vector θ_i, is

$$p(D_i(t_k)|\theta_i, \sigma) = \mathcal{N}\Big(E\big[f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta_i)\big], 2\sigma^2\Big) \tag{5.5}$$

and

$$E\big[f_i(\mathbf{X}^{(i)}(t_{k-1}, \theta_i))\big] = \int_{\Omega_{\mathbf{X}^{(i)}}} f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta_i) \prod_{i=1}^{K_i} \Big[p_i\Big(X_i(t_{k-1})|\hat{X}_i(t_{k-1})\Big)\Big] d\mathbf{X}^{(i)} \tag{5.6}$$

where $\Omega_{\mathbf{X}^{(i)}}$ is the sample space of $\mathbf{X}^{(i)}$, and K_i is the number of chemical species in the expression for f_i. Note that we decide to consider the expectation value of the rate functions (and not the value of the f_is) because we want to mediate the effects of the prensence of random noise on the actual value of the rate functions at each time-point.

While the increments/decrements are conditionally independent given the starting point $X_i\,(t_k)$, the random variables $D_i(t_k)$ are not independent of each other. Intuitively, if $X_i(t_k)$ happens to be below its expected value because of random fluctuations, then the following increment $D_i(t_{k+1})$ can be expected to be bigger as a result, while the previous one $D_i(t_k)$ will be smaller. A simple calculation allows us to obtain the covariance matrix of the vector of increments for the i-th species. This is a banded matrix $\mathbf{C}_i \equiv \mathbf{C} = \mathrm{Cov}(\mathbf{D}_i)$ with all entries zero except for the diagonal elements given by

$$E\Big[D_i^2(t_k) - E[D_i^2(t_k)]\Big] = 2\sigma^2$$

and a non-zero band above and below the diagonal given by

$$E\Big[\big(D_i(t_k) - E[D_i(t_k)]\big)\big(D_i(t_{k-1}) - E[D_i(t_{k-1})]\big)\Big] = -\sigma^2.$$

The likelihood for the observed increments/decrements therefore will be

$$p(\mathbf{D}|\mathbf{\Theta}) = \prod_{i=1}^{N} \mathcal{N}(\mathbf{D}_i|\mathbf{m}_i(\mathbf{\Theta}), \mathbf{C}) = \left(\frac{1}{\sqrt{2\pi\ \det(\mathbf{C})}}\right)^N e^{\sum_{i=1}^{N} -\frac{1}{2}(\mathbf{D}_i - \mathbf{m}_i)^T \mathbf{C}^{-1} (\mathbf{D}_i - \mathbf{m}_i)} \tag{5.7}$$

where $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D}_N\}$, $\mathbf{D}_i = D_i(t_1), D_i(t_2), \ldots D_i(t_M)$ $(i = 1, 2, \ldots, N)$, and $\mathbf{m}_i(t_{k-1}) \equiv E\Big[f_i(\mathbf{X}(t_{k-1}), \theta_i)\Big]$.

Eq. (5.7) can be optimized with respect to the parameters $\mathbf{\Theta} = (\theta_1, \theta_2, \ldots, \theta_N)$ of the model to yield estimates of the parameters themselves and of the noise level. The chief numerical problem of this approach is the computation of the expectations of the rate functions given by equation (5.6). Non-integer values of the coefficients α can make estimating the integral analytically difficult. Hence we use an approximate method in which the Gaussian noise is replaced by an approximate uniform (white) noise, with the amplitude of the uniform noise being obtained as a sample from the Gaussian cumulative distribution function. At the first order, for small σ, we can approximate the Gaussian with zero mean and variance σ with an uniform distribution defined on the interval $[-\frac{\sqrt{2\pi}\sigma}{4}, \frac{\sqrt{2\pi}\sigma}{4}]$, so that

$$\prod_{i=1}^{K_i} p_i = \prod_{i=1}^{K_i} \chi_i \tag{5.8}$$

where

$$\chi_i(X_i) = \begin{cases} \frac{2}{\sqrt{2\pi}\sigma} & \text{if } -\frac{\sqrt{2\pi}\sigma}{4} \leq X_i \leq \frac{\sqrt{2\pi}\sigma}{4} \\ 0 & \text{otherwise.} \end{cases}$$

This approximation makes the calculation of the expectation value of the rate equation (Eq. (5.6)) simpler and reduces the computational time of the procedure. Moreover, experiments reported in [120] demonstrate that it does not influence the accuracy of the parameter estimates until σ is less that 30% of the concentration measurement.
Substituting Eq. (5.8) in Eq. (5.6) gives

$$E[f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta)] = \left(\frac{2}{\sqrt{2\pi}\sigma}\right)^{K_i} \int_{\hat{X}-\frac{\sqrt{2\pi}\sigma}{4}}^{\hat{X}+\frac{\sqrt{2\pi}\sigma}{4}} f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta_i) d\mathbf{X}^{(i)} \tag{5.9}$$

Now, substituting Eq. (5.1) in Eq. (5.9) leads to

$$E[f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta_i)] = \left(\frac{2}{\sqrt{2\pi}\sigma}\right)^{K_i} \left\{ \sum_{h=1}^{N_i} \theta_{ih} \left[\left(\frac{\sqrt{2\pi}\sigma}{2}\right)^{\#(S-S_h)} \times \right.\right.$$
$$\left.\left. \times \prod_{w \in S_h} \frac{1}{\alpha_w + 1} \left(\left(\hat{X}_w + \frac{\sqrt{2\pi}\sigma}{4}\right)^{\alpha_w+1} - \left(\hat{X}_w - \frac{\sqrt{2\pi}\sigma}{4}\right)^{\alpha_w+1} \right) \right] \right\} \tag{5.10}$$

where S is the set containing the indexes referring to all the K_i species appearing in f_i,

and $\alpha_w \neq -1$. In case some orders are equal to -1 Eq. (5.10) takes the following form

$$E[f_i(\mathbf{X}^{(i)}(t_{k-1}), \theta_i)] = \left(\frac{2}{\sqrt{2\pi}\sigma}\right)^{K_i} \sum_{h=1}^{N_i} \theta_{ih} \Bigg\{ \left(\frac{\sqrt{2\pi}\sigma}{2}\right)^{\#(S-S_h)} \times \Bigg[\prod_{w \in S'_h} \frac{1}{\alpha_w+1} \left(\left(\hat{X}_w + \tfrac{\sqrt{2\pi}\sigma}{4}\right)^{\alpha_w+1} - \left(\hat{X}_w - \tfrac{\sqrt{2\pi}\sigma}{4}\right)^{\alpha_w+1} \right) \Bigg] \Bigg[\prod_{w \in S''_h} \ln \frac{\hat{X}_w + \frac{\sqrt{2\pi}\sigma}{4}}{\hat{X}_w - \frac{\sqrt{2\pi}\sigma}{4}} \Bigg] \Bigg\} \quad (5.11)$$

where S'_h is the set of indexes $\{h'_1, h'_2, \ldots, h'_s\}$ such that $\alpha_{h'} \neq -1 \ \forall h' \in S'_h$, and S''_h is the set of indexes $\{h''_1, h''_2, \ldots, h''_s\}$ such that $\alpha_{h''} = -1 \ \forall h'' \in S''_h$.

If in the Eq. (5.7), $\mathbf{m}_i$ is substituted with the expression (5.10) or (5.11), Eq. (5.7) becomes more tractable and can be optimized with respect to the parameters $\mathbf{\Theta} = (\theta_1, \theta_2, \ldots, \theta_N)$ and σ. The values of the model's parameters for which $p(\mathbf{D}|\mathbf{\Theta})$ has a maximum are the most likely values giving the observed kinetics.

5.1.1 Variance of the estimated parameters

To seek the parameter matrix $\mathbf{\Theta}$ that maximizes the function in Eq. (5.7) is equivalent to seeking the parameter matrix $\mathbf{\Theta}$ that maximizes the log-likelihood function given by

$$\ln p(\mathbf{D}|\mathbf{\Theta}) = -\frac{N}{2}\left(\ln(2\pi) \ln(\det\ (\mathbf{C})) \right) - \frac{1}{2} \sum_{i=1}^{N} \left((\mathbf{D}_i - \mathbf{m}_i)^T \mathbf{C}^{-1} (\mathbf{D}_i - \mathbf{m}_i) \right) \quad (5.12)$$

Maximizing the log-likelihood function amounts to minimizing the last term of (5.12) since the other terms do not depend on $\mathbf{\Theta}$. The estimation problem is therefore reformulated as follows:

$$\mathbf{\Theta}^{MLE} = \arg\min_{\mathbf{\Theta}} \sum_{i=1}^{N} \left((\mathbf{D}_i - \mathbf{m}_i)^T \mathbf{C}^{-1} (\mathbf{D}_i - \mathbf{m}_i) \right) \quad (5.13)$$

The maximum likelihood estimate $\mathbf{\Theta}^{MLE}$ has the following appealing asymptotic properties: it is asymptotically unbiased (i.e. $E(\mathbf{\Theta}^{MLE}) = \mathbf{\Theta}^*$, where $\mathbf{\Theta}^*$ denotes the vector of the true values of $\mathbf{\Theta}$), consistent, asymptotically efficient and asymptotically Gaussian [26]. The latter implies that the distribution of the $\mathbf{\Theta}^{MLE}$ converges to a normal distribution with a covariance matrix given by the Cramér-Rao bound that is also the inverse of the Fisher information matrix

$$\mathbf{F}_{\mathbf{\Theta}^{MLE}} = \sum_{i=1}^{N} \mathbf{G}_i^T \mathbf{C}^{-1} \mathbf{G}_i \quad (5.14)$$

where $\mathbf{G}_i$ is called the *sensitivity matrix*.

All the matrices $\mathbf{G}_i$ can be obtained from the sensitivity matrix $\mathbf{S}(t)$ evaluated at the sampling instants. The sensitivity matrix is a $N \times P$ time-dependent matrix, where N is the number of species and p is the length of the parameter vector $\mathbf{\Theta}$. It is defined as follows

$$\mathbf{S}(t) = \frac{\partial \ln \mathbf{m}(t, \mathbf{\Theta})}{\partial \ln \mathbf{\Theta}} = \left(\frac{1}{\mathbf{m}(t, \mathbf{\Theta})} \frac{\partial \mathbf{m}(t, \mathbf{\Theta})}{\partial \mathbf{\Theta}} \right) \Bigg|_{\mathbf{\Theta}=\mathbf{\Theta}^{MLE}} \tag{5.15}$$

The $\mathbf{G}_i$ matrices are obtained from $\mathbf{S}(t)$ as

$$\mathbf{G}_i = [s_i(t_1)^T, \ldots, s_i(t_M)^T]^T$$

where $s_i(t_k)$, $(k = 1, \ldots, M)$, is the i-th row of $\mathbf{S}(t_k)$ $(i = 1, \ldots, N)$, where

$$\mathbf{S}(t_k) = \begin{pmatrix} \frac{\partial \ln m_1(t_k)}{\partial \ln \theta_{i1}} & \cdots & \frac{\partial \ln m_1(t_k)}{\partial \ln \theta_{iN_i}} & \cdots & \frac{\partial \ln m_1(t_k)}{\partial \ln \theta_{NP}} & \\ \frac{\partial \ln m_2(t_k)}{\partial \ln \theta_{i1}} & \cdots & \frac{\partial \ln m_2(t_k)}{\partial \ln \theta_{iN_i}} & \cdots & \frac{\partial \ln m_2(t_k)}{\partial \ln \theta_{NP}} & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \ln m_N(t_k)}{\partial \ln \theta_{i1}} & \cdots & \frac{\partial \ln m_N(t_k)}{\partial \ln \theta_{iN_i}} & \cdots & \frac{\partial \ln m_N(t_k)}{\partial \ln \theta_{NP}} \theta_{NP} & \end{pmatrix}$$

and thus

$$\mathbf{G}_i^T = \begin{pmatrix} \frac{\partial \ln m_i(t_1)}{\partial \ln \theta_{i1}} & \frac{\partial \ln m_i(t_2)}{\partial \ln \theta_{i1}} & \cdots & \frac{\partial \ln m_i(t_M)}{\partial \ln \theta_{i1}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \ln m_i(t_1)}{\partial \ln \theta_{iN_i}} & \frac{\partial \ln m_i(t_2)}{\partial \ln \theta_{iN_i}} & \cdots & \frac{\partial \ln m_i(t_M)}{\partial \ln \theta_{iNi}} \\ \frac{\partial \ln m_i(t_1)}{\partial \ln \theta_{i+1\ 1}} & \frac{\partial \ln m_i(t_2)}{\partial \ln \theta_{i+1\ 1}} & \cdots & \frac{\partial \ln m_i(t_M)}{\partial \ln \theta_{i+1\ 1}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \ln m_i(t_1)}{\partial \ln \theta_{NP}} & \frac{\partial \ln m_i(t_2)}{\partial \ln \theta_{NP}} & \cdots & \frac{\partial \ln m_i(t_M)}{\partial \ln \theta_{NP}} \end{pmatrix}$$

An element $\gamma_{ab}^{(i)}(t)$ of the $\mathbf{G}_i$ $(a = 1, 2, \ldots, M$ and $b = 1, 2, \ldots, N)$, is

$$\begin{aligned} \gamma_{ab}^{(i)}(t) = \left(\frac{1}{\mathbf{m}_i} \frac{\partial \mathbf{m}_i}{\partial \theta_{ib}} \right)_{\mathbf{\Theta}=\mathbf{\Theta}^{MLE}} = \left(\frac{1}{\mathbf{m}_i} \right)_{\mathbf{\Theta}=\mathbf{\Theta}^{MLE}} \left(\frac{\sqrt{2\pi}\sigma}{2} \right)^{-\#(S_b)} \\ \times \prod_{w \in S_b} \left[\left(\hat{X}_w(t) + \frac{\sqrt{2\pi}\sigma}{4} \right)^{\alpha_w+1} - \left(\hat{X}_w(t) - \frac{\sqrt{2\pi}\sigma}{4} \right)^{\alpha_w+1} \right] \frac{1}{\alpha_w + 1} \end{aligned} \tag{5.16}$$

if $\alpha_w \neq -1$ $\forall w \in S_h$.

The square root of the p-th diagonal element of $\mathbf{F}_{\mathbf{\Theta}^{MLE}}^{-1}$ gives an estimate of the standard deviation of the p-th component of $\mathbf{\Theta}$ [49].

5.1.2 Parameter space restriction

The search for the optimal values of rate constants can be made more efficient if we provide the algorithm of optimization of Eq. (5.7) with the initial guesses for these constants. In this way the algorithm does not waste time in exploring large regions of the

parameter space in which the model in Eq. (5.2) is not valid. For this purpose we include in our framework a procedure for the automatic calculation of the initial guesses of the parameters. Therefore, the task to direct the inference method to efficiently exploring the parameter space is not left to the user, who often does not have a precise idea about a reasonable value of the parameters.

The derivatives dX/dt at all measured time points t_k can be interpreted as slopes [191, 169]. Given the species i (with $i = 1, \ldots, N$), we can estimate these slopes from the data as $s_i(t_k)$, and approximate the differential equations as

$$s_i(t_k) \approx \left.\frac{dX_i}{dt}\right|_{t=t_k} \tag{5.17}$$

If the data consist of N species and the concentration of each species i is measured at M time points $(X_i(t_1), X_i(t_2), \ldots, X_i(t_M))$, we estimate $M \times N$ slopes $s_i(t_k)$ $(k = 1, \ldots, M)$. In fact, for each species we have M differential equation of the form

$$s_i(t_k) \approx f_i(X_1(t_k), X_2(t_k), \ldots, X_N(t_k); \theta_{i1}, \theta_{i2}, \ldots, \theta_{iN_i}) \tag{5.18}$$

that form a system of M algebraic equations with $M \times N_i$ unknown variables θs, as the slopes s are measurable from the data. In general $M \neq N_i$ and more often $M \gg N_i$ so that the system of $M \times N_i$ equation results overdetermined. The prediction intervals for the parameter can thus be obtained by computing the solution of the system (5.18) with the least squares procedure for overdetermined systems. Note that at this stage we are not interested in a very precise estimate of the rate constants, but only in an approximate guess. Note also that the least squares method should be considered only as a method of fitting a line to a set of data, not as a method of statistical inference. The parameters calculated with the method of least squares should be called least squares *solutions* rather than least squares *estimates*, because they are the solutions of the mathematical problem of minimizing the residual sum of squares rather then estimates derived from a statistical model. However, if we make the assumption that the residuals are normally distributed and independent with the same variance σ^2, then the maximum likelihood approach to the estimation of model parameters from data yields the classical formulae for least squares.

A system of equations similar to the system (5.18) can be written also for the experimental uncertainties Δs_i affecting the slopes s_i:

$$\Delta s_i(t_k) \approx \Delta f_i(X_1(t_k), X_2(t_k), \ldots, X_N(t_k); \theta_{i1}, \theta_{i2}, \ldots, \theta_{iN_i}) \tag{5.19}$$

where

$$\Delta s_i = \Delta\Big[\theta_{i1} \prod_{j \in S_1 \subseteq [1,N]} X_j^{\alpha_j}\Big] + \Delta\Big[\theta_{i2} \prod_{j \in S_2 \subseteq [1,N]} X_j^{\alpha_j}\Big] + \cdots + \Delta\Big[\theta_{iN_i} \prod_{j \in S_{N_i} \subseteq [1,N]} X_j^{\alpha_j}\Big] \tag{5.20}$$

By using the standard formulas of the error propagation, a single term of the sum on the right-hand side of Eq. (5.20) is

$$\frac{\Delta\left[\theta_{i1}\prod_{j\in S_1\subseteq[1,N]}X_j^{\alpha_j}\right]}{\left|\theta_{i1}\prod_{j\in S_1\subseteq[1,N]}X_j^{\alpha_j}\right|}=\frac{\Delta\theta_{i1}}{\theta_{i1}}+\frac{\Delta\left[\prod_{j\in S_1\subseteq[1,N]}X_j^{\alpha_j}\right]}{\left|\prod_{j\in S_1\subseteq[1,N]}X_j^{\alpha_j}\right|}=\frac{\Delta\theta_{i1}}{\theta_{i1}}+\sum_{h=1}^{\#S_1}|\alpha_h|\frac{\Delta X_h}{|X_h|}$$

where $\#S_1$ is the cardinality of the set S_1. Therefore, Eq. (5.20) becomes

$$\Delta s_i=\sum_{\nu=1}^{N_i}\left\{\left(\frac{\Delta\theta_{i\nu}}{\theta_{i\mu}}+\sum_{h=1}^{\#S_\nu}|\alpha_h|\frac{\Delta X_h}{|X_h|}\right)\cdot\left|\theta_{i\mu}\prod_{j\in S_\nu\subseteq[1,N]}X_j^{\alpha_j}\right|\right\} \tag{5.21}$$

By assuming that the measurements of times are not affected by errors, the error Δs_i is calculated from Eq. (5.2) as follows

$$\Delta s_i(t_k)=\frac{1}{t_k-t_{k-1}}\Big(\Delta X_i(t_k)-\Delta X_i(t_{k-1})\Big)$$

where $\Delta X_i(t_k)$ is the experimental error on the measurement of concentration of species i at time t_k. Therefore $\Delta s_i(t_k)$ can be obtained from the data, and the system (5.21) can be solved to find the size of the prediction intervals $\Delta\theta$ of θs with the same procedure used for the system (5.18). These intervals are also upper-bound measures of the errors that propagate to the rate constants from the concentration measurements.

5.1.3 Parameter inference in not - mass action models

Although the generalized mass action law (Eq. (5.1)) is able to describe a wide variety of biochemical processes, this empirical rate equation is not the only principle for modelling continuous time nonlinear dynamics. The other two main principles are the Michaelis-Menten kinetics and the Hill kinetics. The Michaelis-Menten kinetic is derived from the mass action law when the steady-state assumption is verified. The steady-state approximation assumes that the concentration of any intermediate in the reaction is unchanged (the rate of the intermediate production is equal to the rate at which it is used). In an enzymatic catalysis, for instance, this assumption is verified if the concentration of the substrate is much larger than the concentration of the enzyme. If the steady-state approximation can be applied, i.e. when the input values of the time series are such to determine a Michaelis Menten kinetics, the parameters inferred from a generalized mass action analytical form can be combined to calculate the values of the Michaelis-Menten constants.

While the generalized mass action law can still handle Michaelis-Menten approximation, the kinetics described by the Hill function deserves a different mathematical treat-

ment before the inference method proposed in this work can be applied. Consider the general form of the Hill function $h(x)$, where x is the species concentration.

$$h(x) = \theta \frac{x^n}{c^n + x^n} \tag{5.22}$$

where the constants $c, \theta \in \mathbb{R}$ and $n \in \mathbb{N}$.

The Hill function produces a sigmoidal curve that can be fitted to a Gaussian error function (the so-called `erf` function) as in the following:

$$h(x) \cong c_1 \left[\texttt{erf}(c_2 x^{c_3} + c_4) \right] + c_5 \tag{5.23}$$

where c_1, and $c_5 \in \mathbb{R}$ are constants, and $c_2 \cong c_3 \cong \sqrt{n}$ and $|c_4|$ is of the same order of magnitude as the variance affecting the experimental concentration time series of the species x. Therefore, our method can fit a Hill form to an `erf` function and optimize the `erf` function with respect to the parameters c_1 and c_5. Many tests both on synthetic and real data have shown that the Gauss error function calculated with the parameter values obtained by our procedure for c_1, and c_5, and $c_2 \cong \sqrt{n}$ and $c_4 \cong \sigma_{exp}$, is a good fit of the Hill function, since it fits the maximum, minimun and the point of inflexion of the Hill sigmoid. Thus, the value of x for which the `erf` function presents an inflexion is an estimate of the constant c in the Hill form in Eq. (5.22). For our preliminary analysis of the tratment of Hill kinetics, the user is asked to specify the value of the Hill coefficient n, because as for the partial reaction orders αs in the model (5.1), it is not included in the optimization procedure. Therefore, once the Hill equation is entered, it is easy to automatically introduce in the equation system a new species, say X_{erf}, whose rate of change in time is calculaded as follows

$$\frac{X_{erf}(t)}{dt} = \texttt{erf}(\sqrt{n} x(t)^{\sqrt{n}} + \sigma_{exp})$$

where $x(t)$ is the experimental time series of concentrations of species x. Then, the procedure described in the previous section can estimate parameters c_1 and c_5 that optimize the fit with the experimental observations. Using this method, the mathematical scheme of complex reaction kinetics (i.e. not-mass action Hill kinetics) could be handled. If, for instance, the rate equation of the i-th species contains mass action terms and Hill terms

$$\frac{dX_i}{dt} = \sum_{h=1}^{N_i - 1} \left(\theta_{ih} \prod_{w \in S_h} X_w^{\alpha_w} \right) + \theta_{in_i} \frac{{X_{N_i}}^n}{c^n + X_{N_i}^n}$$

the equation can be approximated as follows

$$\frac{dX_i}{dt} \cong f_i(\mathbf{X}^{(i)}(t); \theta_i) \cong \sum_{h=1}^{N_i - 1} \left(\theta_{ih} \prod_{w \in S_h} X_w^{\alpha_w} \right) + c_1 X_{erf}^{(i)} + c_5 \tag{5.24}$$

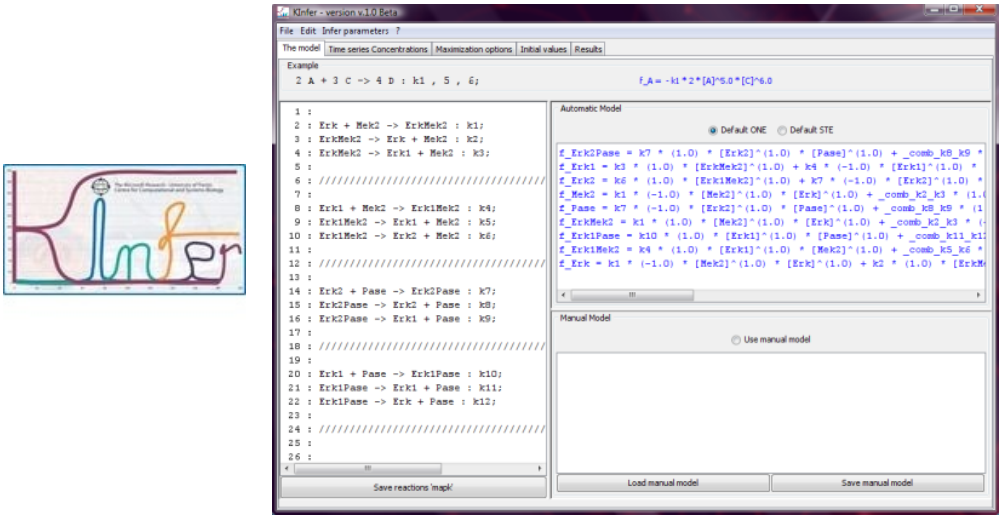
where

$$\frac{dX_{erf}^{(i)}}{dt} = \left[\texttt{erf}(c_2 X_{N_i}^{c_3} + c_4)\right]$$

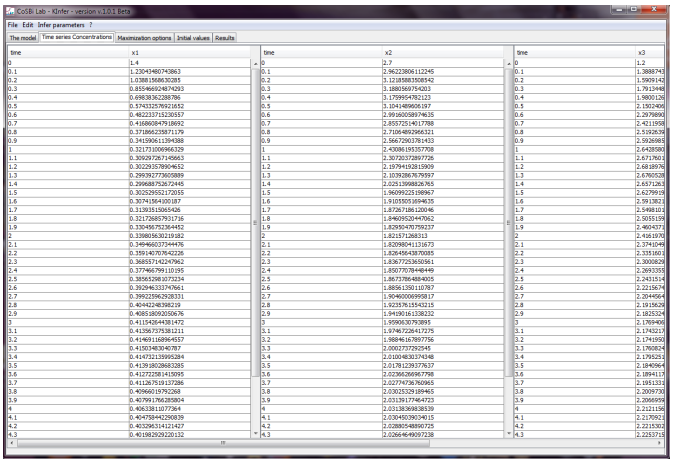
is numerically calculated from the experimental time-series of $X_{N_i}(t)$ and $c_2 \cong c_3 \cong \sqrt{n}$ and $|c_4| \cong \sigma_{exp}$. In this way, Eq. (5.24) has still a mass-action form.

5.2 The inference tool: KInfer

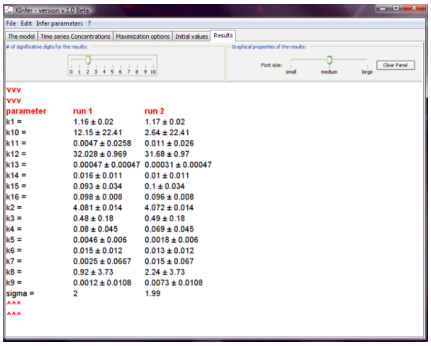
We developed the prototype software, called KInfer (Kinetic Inference), that implements the mathematical procedure described in the previous section. In Figure 5.1, some screenshots of the current version of the tool are shown.



(b)



(c)



(d)

Figure 5.1: Screenshots of KInfer, a tool for inferring parameters of a biochemical system.

KInfer is a software tool written in Java, distributed as a jar package and downloadable at `http://www.cosbi.eu/index.php/research/prototypes/kinfer`. KInfer requires Java 2 Runtime Environment (JRE) version 5 or newer, or an equivalent JRE. KInfer relies upon the JAMA library, a cooperative product of the MathWorks and the National Institute of Standards and Technology (NIST) [128]: this library is installed within the KInfer directory when the user installs the software.

The whole architecture of KInfer consists of seven main blocks (Figure 5.2): two manage the data needed as input (i.e. the model and the time series), one is responsible for automatically generate the general mass action model, one calculates initial guesses for the parameters, one builds the probability density function properly combining the model structure and the time series data, one performs the maximization step and finds the parameter which maximize the previously built probability density function and finally the last module represent the output interface that show the results.

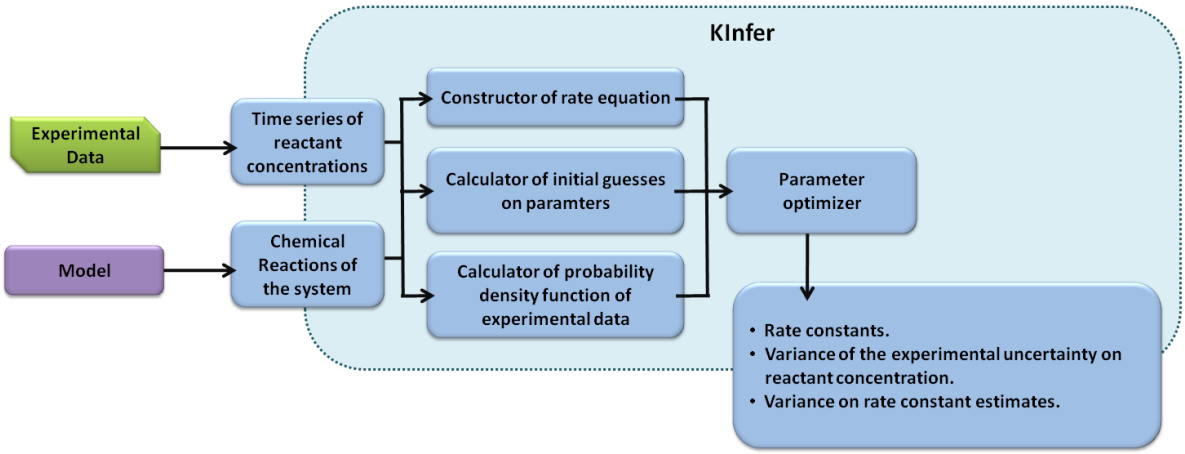


Figure 5.2: Graphical representation of the general architecture of KInfer.

In the rest of this section we will explain in more details how each of the modules works.

The first input needed by the software is the set of chemical reactions defining the model.

In the left part of Figure 5.1(b) the user can input the set of chemical reactions describing the kinetics of the system, with the following syntax:

$$a\ A + b\ B + \cdots \rightarrow a'\ A' + b'\ B' + \cdots :\ k,\ \alpha,\ \beta, \ldots ;$$

On the left-hand side of the arrow, the reactants $(A, B, \ldots)$ and the reactants stoichiometric coefficients $(a, b, \ldots)$ are indicated (separated by an empty space). On the right-hand side of the arrow the products $(A', B', \ldots)$ and the product stoichiometric co-

efficients $(a', b', \dots)$ are indicated. After the list of products, the user has to specify the name for the rate constant (k in the example above) associated to the reaction, followed by a comma-separated list of the partial orders of reaction $(\alpha, \beta, \dots)$ for that reaction. The specification of the reaction must end with a semicolon.

The reaction orders are supposed to be given and do not undergo any inference procedure. This list of partial orders of reaction is optional: if not specified, the default value can be chosen by the user between 1 (that is the option "Default ONE") or the stoichiometric coefficients associated with the species (that is the option "Default STE"). While the user is typing, KInfer automatically translates the list of chemical reactions into the corresponding rate equations following the General Mass Action kinetic format and shows the result in the upper right part of the GUI (see Figure 5.1(b)). In the same part of the graphical user interface typing errors that prevent the tool from building the correct mass action model are reported to the user.

Using the proper button or menu options, it is possible to save a model into a textual file for future use and then load it again into KInfer using the classical save/open dialog windows that allow the user to browse the filesystem and open the input textual file defining the model.

Alternatively to the automatically generated model, the user is allowed to input a manual model, through the "Manual Model" part of the interface (Figure 5.1(b)). The user can enter an ordinary differential equation model without specifying the reactions' list in the standard chemical notation: the syntax of the ODE model has to be the same as the one of the automatically generated model (some examples are presented in Chapter 6). The manual model can be used to specify general mass action laws with possibly real numbers as partial order of reaction: in this first prototype version of the software, it is not possible to specify a general kinetic law, but we plan to add this feature in the future (see Chapter 7 for discussions about this topic).

Along with the specification of the set of reactions involved in the system, KInfer requires the experimental time series data of the concentration (or number of molecules) of the species in the system.

In order to load those data, the user has to select the "Load concentrations..." item from the "File" menu. This will show a classical open dialog window that can be used to open the directory containing the concentrations file. This file has to be a Comma Separated Values (CSV) file, whose first row is the list of all the reactants $sp_1, \dots, sp_N$ contained in the model, preceded by the keyword "time". The following rows of the file

need to be the concentration levels expressed by real numbers as follows:

$$
\begin{aligned}
& t_0, [sp_1]_{t_0}, [sp_2]_{t_0}, \ldots, [sp_N]_{t_0} \\
& \ldots \\
& t_i, [sp_1]_{t_i}, [sp_2]_{t_i}, \ldots, [sp_N]_{t_i} \\
& \ldots \\
& t_M, [sp_1]_{t_M}, [sp_2]_{t_M}, \ldots, [sp_N]_{t_M}
\end{aligned}
$$

The loaded concentration file is then displayed in the "Time series concentration" tab as shown in 5.1(c). In this version of KInfer there is not the possibility of modifying those data from the interface, but its a very simple modification that can be easily implemented in the next version of the software.

With the information provided in the first two tabs of the graphical interfaces, KInfer internally creates and holds in appropriate data structures the probability density function in Eq. (5.7): as this mathematical formula is parametric with respect to the kinetic values driving the biochemical system, it is used to find the set of kinetic parameters that maximize its value.

The optimization algorithm included in KInfer is a Genetic Algorithm (GA) [78]. This choice has been driven by the fact that a biological model of realistic size and complexity presents a high number of parameters with possible nonlinear relations between them. A GA is a population based stochastic optimization technique, that, starting from a set of initial guesses about the solution, determines the next set of possible solutions to the optimization problem on the basis of the results obtained from the preceding set and approaching step by step the best solution. In particular, in the GA approach, the evolution starts from a population of randomly generated individuals. Then in each generation the fitness of every individual in the population is evaluated and multiple individuals are stochastically selected from the current population (based on their fitness). The chosen individuals are modified (recombined and possibly randomly mutated) to form a new population. The new population will be used in the next iteration of the algorithm. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. The selection operation involves the evaluation of each possible solution with respect to the target assigned: the lower the log-likelihood value is, the better the solution is considered. The next step is to select the solutions for the next generation in such a way that those with higher fitness have higher probability of selection: to each guessed solution will be assigned a selection probability derived by the ratio of its square fitness and the sum of the squared fitness of all the solutions. The selected solutions are then subjected to

cross-over, mutation and innovation operators. To realize cross-over, every two parents create two children in the following way: the algorithm selects randomly from the first parent how many and which parameters will have to be kept in the first child. Then from the second parent the algorithm takes the complementary number of variables and uses these values to complete the first child. The second child is then built with the remaining variables of the two parents. The mutation operation, with a low probability (in our examples p = 0.1), randomly selects one variable to be mutated. After the selection, the value of the variable is changed selecting (again randomly) from the possible values it can take excluding the currently one. Finally, the innovation operator randomly selects new solutions never tested to be performed. Usually this operator is kept at low rate (here at 5%), trying to optimize the trade-off between exploration and exploitation. Once the new population of experiments is derived from the algorithm, it is then proposed as a new generation for the next algorithm iteration. The size of each population of solution in each generation is maintained constant. These methods have been designed primarily to address problems that cannot be tackled through traditional optimization algorithms. Such problems are characterized by discontinuities, lack of derivative information, noisy function values and disjoint search spaces.

In the KInfer graphical interface, the user (in the "Maximization options" tab) can modify the main parameters of the method which have the following interpretation:

- *Multiplier*: a real number, that is used to perform subsequent run of the algorithm on the same objective function but with different initial value ranges for the unknown parameter. E.g. using a 0.5 multiplier means that for each complete run of the GA, the range of the initial parameters search space is halved.
- *NTrial*: is the number of different runs of the algorithm that are performed applying, each time, the *Multiplier* parameter to the parameters bounds
- *NGenerations*: is the maximum number of generations of the GA
- *Nexp*: is the size of each population in a single step of the GA
- *Crossed*: is the number of population elements that are target of the cross-over operation between two steps of the GA. This number has to be lower than *Nexp* and it has to be even.

In order to limit the search space of the optimization algorithm, as explained in Section 5.1.2, we implemented a procedure for the automatic calculation of the initial guesses of the parameters. Through the "Initial values" tab the user can take advantage of this procedure by simply clicking the "Calculate!" button in the upper part of the window. This will calculate the Stineman value for each of the parameter indicated in the model,

using the time series concentration loaded before and considering the fact that each value has a measurement error associated (see Section 5.1.2 for a detailed description of the procedure).

This whole process can be avoided by the user that already knows a possible range of variability for each parameter: in this case the user can simply load those data from a textual file written as follows:

```
k1 => lower: 1.0E-8 upper: 1.0
k2 => lower: 1.8 upper: 10.0
k3 => lower: 0.27 upper: 1.0
sigma => lower: 1.0E-8 upper: 1.0
```

The user can also dynamically modify each range value through the graphical interface: it is enough to select from the list the parameter that has to be changed, and then type the new range in the two textfields on the right part of the window. To save those values for the current run of the inference algorithm, the user has to click the "Update ranges initial parameters!" button. To save them for future runs the user can click the "Save initial values..." button on the left, and then select a file in which those values will be stored in the format shown above.

Finally, as soon as the model, the concentration data, the maximization algorithm and the initial values are loaded and correctly set up, the inference algorithm can be executed just selecting the option "Infer!" from the "Infer parameters" menu.

After the calculations, that can take some time depending on the number of parameters that has to be inferred and the choices made for the maximization algorithm, the results are listed in the "Results" tab (Figure 5.1(d)). In the textarea, the user can find the list of all the parameters with an inferred value for each of the runs of the GA that he/she decided to make. The user can select the results from the textarea and simply copy/paste them, according to his/her needs.

5.3 Some concluding considerations

The results of the application of our inference procedure to the calibration of models of synthetic and real biochemical networks (see next chapter) show that the method converges to the expected solutions within the bounds of the experimental errors that propagates from concentration measurements to the kinetic rate constants. The chosen method differs from the classical fitting procedure because it relies on the construction of a model of probability that is intrinsically able to handle the noise contained in dataset coming from real experiments on biological systems. In Section 6.5 we will give further details about the comparison of the method presented in this work and other existing

methods and tools for parameter estimation. Here we want just to list some important features missing from the existing methods that are present instead in ours. The first is the automatic computation of the initial guesses of the parameters. In this way, the user is not forced to insert any *a priori* knowledge about the system, that often is quite hard to find. At the same time, the method is equipped with a rigorous procedure referring only to the experimental concentration measurements to identify a region of the parameter space where the optimization of the probability density function takes place. The second feature is the implementation of the experimental error propagation. The evaluation of the experimental uncertainty on the rate constants estimates is particularly useful if the parameter inference is incorporated in projects of experimental design. If big error values are found, it may be a signal that some inconsistencies between the structure of the model and the input data have been identified; this inconsistency can be due to the fact that the input dataset is not accurate enough for inferring the rate constants of the system (i.e. new experimental time series need to be collected), but it can also point out that the model in itself is not structurally able to account for the experimental measures. If more than one dataset are all consistent with one other and the accuracy of the measure is good, then a change of the model is needed; otherwise if the structure of the model is well validated and trusted, just more (or different) experiments need to be performed.

As a final remark, it is important to point out that the software prototype implementing the inference procedure described in previous sections can be used for interfacing the outcomes of the wet-lab activity for the concentration measurements with the softwares for modelling and simulation of biochemical networks: actually KInfer is part of the CoSBiLab platform (`http://www.cosbi.eu/index.php/research/prototypes/overview`) and can be seen as a essential tool in the overall objective of creating an artificial laboratory in which it is possible to replicate in-silico the activities that are usually performed in real wet labs. In particular KInfer can feed BlenX models with the numerical kinetic parameter needed for performing stochastic simulations of the system under consideration. For discussions about the integration of the two different frameworks, see Chapter 7.

Chapter 6

Applying KInfer on biochemical case studies

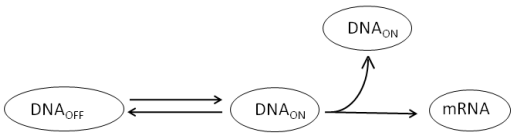
In this chapter we report and discuss the application of KInfer to case studies of increasing complexity. These case studies include first and second order chemical reactions, didactical examples of biochemical networks, and more complex biological pathways like small scale genetic network with feedback loops, cell cycle regulation mechanisms and the NF-kB pathway. Case studies with both synthetic and experimental datasets have been chosen and all the results shown in this chapter have been presented in different pubblications by the author of this thesis and colleagues [120, 119, 118].

Discussions about similar results that can be obtained with other available software tools for parameter inference are presented in the last section of this chapter.

6.1 Gene transcription and transcriptional regulation

In this example we consider the transcription of a single gene as given by the model of Golding et al. in [80]. The DNA for the tagged mRNA is switched on and off by polymerase binding and unbinding, respectively. Only polymerase-bound DNA is transcribed into mRNA. The system is depicted in Figure 6.1.

We set the initial conditions $DNA_{OFF} = 1$, $DNA_{ON} = 0$ and $mRNA = 0$, and we generated a set of 100 data points at temporal resolution of 1. Typical measurements units are "number of molecules" for the species amount and "minutes" for time. Our estimates of parameters are reported in Table 6.1; the comparison of the estimated and experimental system's behavior in Figure 6.2 shows a strong agreement. The accuracy of the results is comparable with one of those obtained by Reinker et al. [163] for the same network.



$$DNA_{OFF} \xrightarrow{\theta_1} DNA_{ON}$$
$$DNA_{ON} \xrightarrow{\theta_2} DNA_{OFF}$$
$$DNA_{ON} \xrightarrow{\theta_3} DNA_{ON} + mRNA$$

Figure 6.1: Golding's model of gene transcription process.

Parameter	Actual value	Initial guesses	Estimated value
θ_1	0.027	[0.0242; 0.0249]	0.0244 ± 0.0007
θ_2	0.1667	[0.151; 0.152]	0.152 ± 0.001
θ_3	0.4	[1.578; 2.385]	1.579 ± 0.807
σ	0.5	[0; 1]	0.445

Table 6.1: Estimated parameter values for the Golding's model of gene transcription (Figure 6.1).

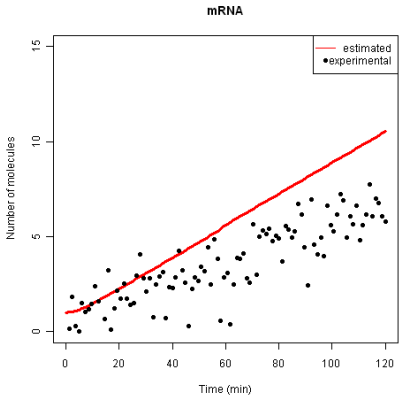
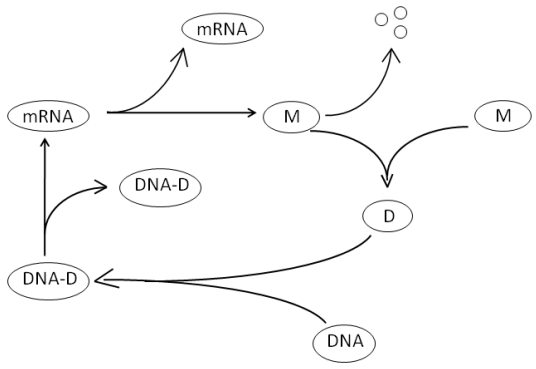


Figure 6.2: Time series of gene transcription case study. The actual and the estimated time series of molecules of mRNA in the model network shown in of Figure 6.1.

We have also considered a more complex model: the Goutsias model of gene transcription regulation [83, 163]. Figure 6.3 illustrates this model. The mRNA is translated into a protein monomer M that can dimerise. The dimer D, in turn, can bind to its DNA and acts as a transcription factor to auto-regulate its own mRNA production. Both mRNA and protein are degraded at constant rates. The set of reactions of this network is reported in Figure 6.3. As in [163], we used this set of reactions to generate a synthetic dataset of the time series of the number of molecules for each component in the system. The dataset contains of 100 data points at the time resolution of 1.2 min. As initial values we used M = 2, D = 4, DNA = 2, and mRNA = 0, DNA·D = 0. All the reaction constants are in units of per seconds.

Table 6.2 reports the estimates of the rate constants, that within the estimated error ranges, are in agreement with the actual values and with the results obtained by [83].



$$mRNA \xrightarrow{\theta_1} mRNA + M;\ M \xrightarrow{\theta_2} \emptyset$$
$$DNA \cdot D \xrightarrow{\theta_3} mRNA + DNA \cdot D$$
$$mRNA \xrightarrow{\theta_4} \emptyset;\ DNA + D \xrightarrow{\theta_5} DNA \cdot D$$
$$DNA \cdot D \xrightarrow{\theta_6} DNA + D;\ 2M \xrightarrow{\theta_7} D;\ D \xrightarrow{\theta_8} 2M$$

Figure 6.3: Goutsias's model of gene transcription regulation.

Figure 6.4 shows the actual and the estimated dynamics.

Parameter	Actual value	Initial guesses	Estimated value
θ_1	0.043	[0.01; 0.08]	0.042 ± 0.007
θ_2	0.0007	[0.0001; 0.001]	0.0004 ± 0.0009
θ_3	0.715	[0; 1]	0.1051 ± 0.1
θ_4	0.00395	[0.00340; 0.00386]	0.0038 ± 0.0005
θ_5	0.02	[0.01; 0.04]	0.019 ± 0.03
θ_6	0.4791	[0; 1]	0.62 ± 0.17
θ_7	0.083	[0.01; 0.2]	0.12 ± 0.02
θ_8	0.5	[0; 1]	0.7 ± 0.1
σ	1	[0,2]	0.95

Table 6.2: Estimated parameter values for the Goutsias model of trascriptional regulation (Figure 6.3).

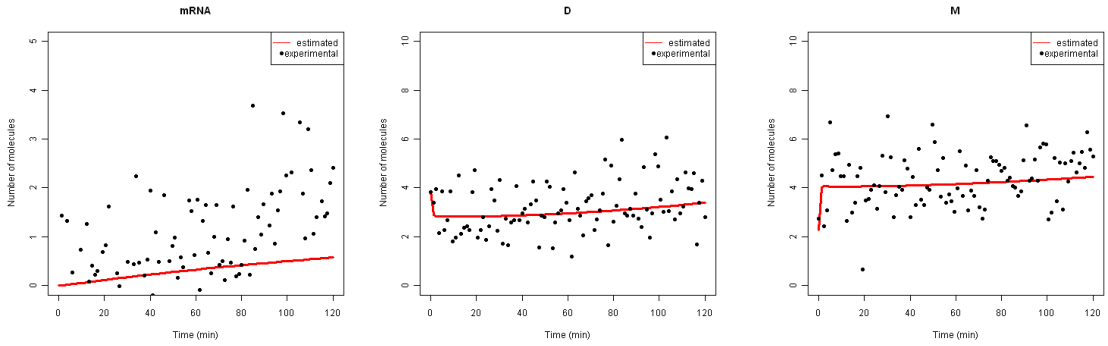


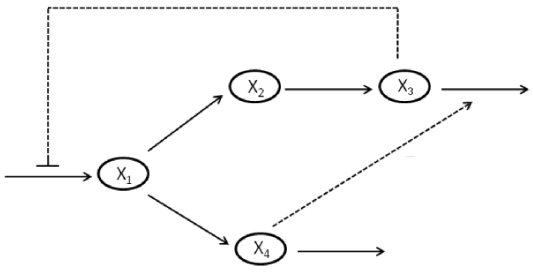
Figure 6.4: Estimated and experimental time series of mRNA, dimer (D), and monomer (M) in Goutsias's model of gene transcription.

6.2 A didactic example of biochemical network

The system depicted in Figure 6.5 is representative of a small biochemical network of 4 interacting species. The network has two feedback loops:

- the species X_3 inhibits the production of species X_1;
- the species X_4 promotes the activation of X_5, inducing X_3 degradation.

A numerical implementation with typical parameters is given by the set of ordinary differential equations in Figure 6.5. This system of equations has been used to create the artificial time series of 51 data points with a time resolution of 0.2. Typical units might be mM for the concentration and minutes for time, but the example could as well run on an hourly scale and with variables of different nature. Table 6.3 lists the results and Figure 6.6 shows the dynamic simulations. Within the experimental uncertainties, these results are in agreement with the expected ones and with those in [33]. The peculiar feature of this example is that the partial orders of reaction are non-integer, so the manual model option of KInfer has been used in order to be able to obtain the estimated results.



$$
\begin{aligned}
X_1' &= \theta_1 X_3^{-0.8} - \theta_2 X_1^{0.5} & X_1(t_0) &= 1.4 \\
X_2' &= \theta_3 X_1^{0.5} - \theta_4 X_2^{0.75} & X_2(t_0) &= 2.7 \\
X_3' &= \theta_5 X_1^{0.75} - \theta_6 X_3^{0.5} X_4^{0.2} & X_3(t_0) &= 1.2 \\
X_4' &= \theta_7 X_1^{0.5} - \theta_8 X_4^{0.5} & X_4(t_0) &= 0.4
\end{aligned}
$$

Figure 6.5: A didactic example of biochemical network with four variables and the system of ordinary differential equations describing it.

Parameter	Actual value	Initial guesses	Estimated value
θ_1	12	[10.18; 13.84]	11.37 ± 3.66
θ_2	10	[8.28; 11.74]	9.39 ± 3.46
θ_3	8	[9.81; 9.87]	9.83 ± 0.06
θ_4	3	[3.92; 3.99]	3.98 ± 0.07
θ_5	3	[2.91; 2.96]	2.94 ± 0.05
θ_6	5	[4.89; 4.91]	4.90 ± 0.02
θ_7	2	[1.50; 2.55]	1.84 ± 1.05
θ_8	6	[4.01; 8.17]	5.5 ± 4.16
σ	0.1	[0.1; 0.3]	0.3

Table 6.3: Estimated parameter values for the network in Figure 6.5

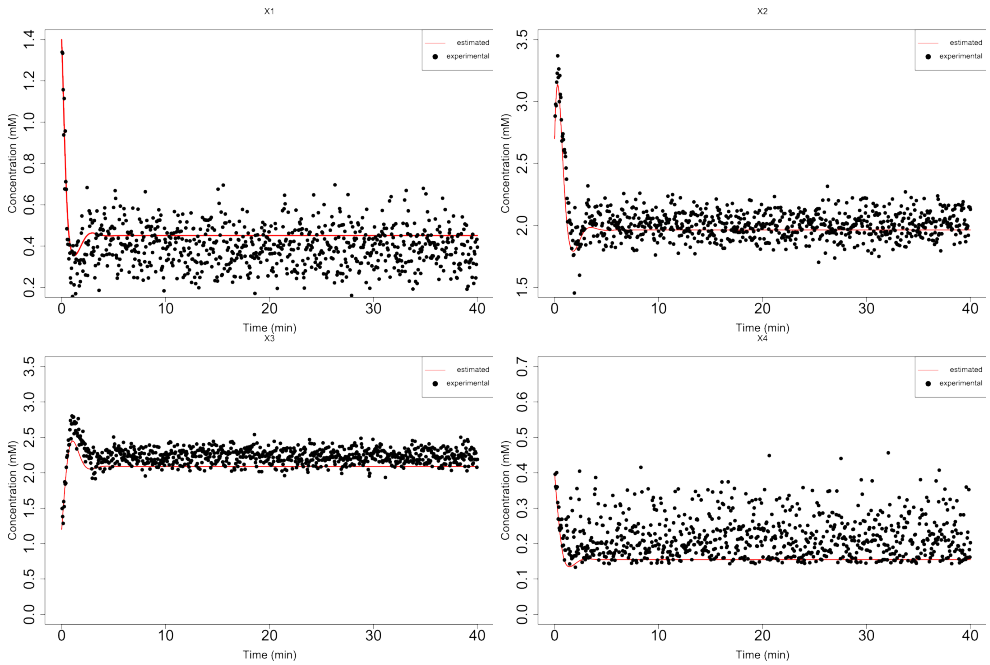


Figure 6.6: Estimated and experimental time series of the species X_1, X_2, X_3, and X_4 in the network of Figure 6.5

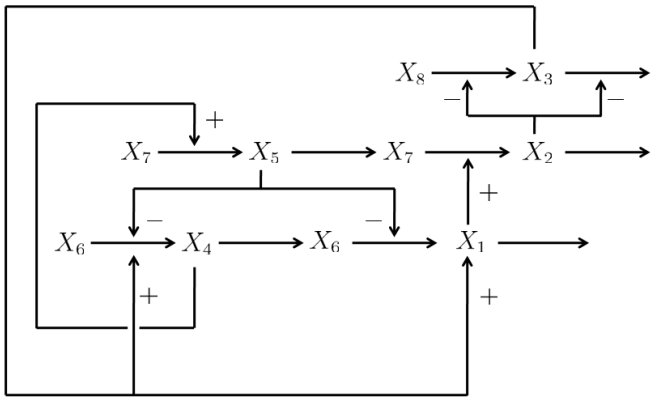
6.3 A regulatory genetic network

To confirm the validity of our inference model we also considered the calibration of a typical model of a small-scale gene network shown in Figure 6.7. Because of the presence of positive and negative loops and its non-linear dynamics, this model is considered a challenging benchmark for comparative tests and analysis of inference methods. Here we report a brief description of the model and we refer the reader to the studies of Hlavacek and Savageau [94], Savageau [167, 168] and Kikuchi et al. [104] for the details of the biological and mathematical aspects of this regulatory network.

Figure 6.7 is a diagrammatic representation of the gene interaction system originally proposed by Hlavacek and Savageau [94] to analyze the interaction of regulator and effector genes. The regulator gene encodes a protein that acts at the level of transcription to bring about induction, and the effector gene encodes an enzyme that catalyzes a pathway in which the inducer of the system is an actual or functional intermediate [94]. The regulator can negatively or positively influence transcription at the promoter of each gene, and these influences can be respectively facilitated or antagonized by inducer. This gene network model is an idealized regulatory system. Nevertheless it captures the essential features of many actual systems [94, 167, 168].

The network consists of two genes (gene 1, called *effector* and gene 4, called *regula-*

tor). The state of the system is described by five variables $X_1, \ldots, X_5$. X_1 is an mRNA produced from gene 1, X_2 is an enzyme protein it produces, and X_3 is an inducer protein catalyzed by X_2. X_4 is an mRNA produced from gene 4 and X_5 is a regulator protein it produces. Positive feedback from the inducer protein X_3 and negative feedback from the regulator protein X_5 are assumed in the mRNA production of gene 1 and 4. These five variables represent quantities that vary during induction and are determined by the processes of transcription, translation, specific degradation, dilution, and metabolism. The variables X_6 and X_7 denote the precursor pools for mRNA and protein biosynthesis, respectively. In our model, as in [104] they are maintened constant with respect to time. Finally, X_8 denotes a substrate concentration representing an independently determined environmental signal to which the system responds.



$$\frac{dX_1}{dt} = \alpha_1 X_3 X_5^{-1} - \beta_1 X_1^2$$
$$\frac{dX_2}{dt} = \alpha_2 X_1^2 - \beta_2 X_2^2$$
$$\frac{dX_3}{dt} = \alpha_3 X_2^{-1} - \beta_3 X_2^{-1} X_3^2$$
$$\frac{dX_4}{dt} = \alpha_4 X_3^2 X_5^{-1} - \beta_4 X_4^2$$
$$\frac{dX_5}{dt} = \alpha_5 X_4^2 - \beta_5 X_5^2$$

Figure 6.7: Genetic network model of Hlavacek and Savageau [94].

The time courses of $X_1, \ldots, X_5$ were artificially prepared solving the S-system in Figure 6.7 with the numerical integrator XPPAUT [61] with the parameter values and initial conditions listed in Table 6.4.

$\alpha_1 = 5$	$\beta_1 = 10$	$X_1(0) = 0.1$
$\alpha_2 = 10$	$\beta_2 = 10$	$X_2(0) = 0.12$
$\alpha_3 = 10$	$\beta_3 = 10$	$X_3(0) = 0.70$
$\alpha_4 = 8$	$\beta_4 = 10$	$X_4(0) = 0.70$
$\alpha_5 = 10$	$\beta_5 = 10$	$X_5(0) = 0.18$

Table 6.4: Parameters that determine the dynamic action of the S-system in Figure 6.7. These values are expressed in arbitrary units and were determined artificially by Kikuchi et al. ([104]) in order to realize the network in Figure 6.7.

Parameter	Value	$\Delta\alpha$	σ_α
α_1	5.41	0.01	0.31
α_2	7.91	0.01	0.76
α_3	2.77	0.01	~ 0
α_4	9.61	0.01	0.27
α_5	10.68	0.01	0.92

Parameter	Value	$\Delta\beta$	σ_β
β_1	11.12	0.01	0.28
β_2	7.691	0.005	0.4
β_3	2.75	0.005	~ 0
β_4	11.84	0.01	0.42
β_5	10.992	0.002	0.81

Table 6.5: Estimates of the kinetic parameters of model in Figure 6.7.

Table 6.5 presents the parameter estimates obtained by KInfer and graphs in Figure 6.8 show the simulation curves (solid lines) obtained by solving system in Figure 6.7 with the estimated parameters. The inferred rate constants are comparable with those used to generate the input data in Table 6.4 and reproduce the expected time series (black points in graphs of Figure 6.8).

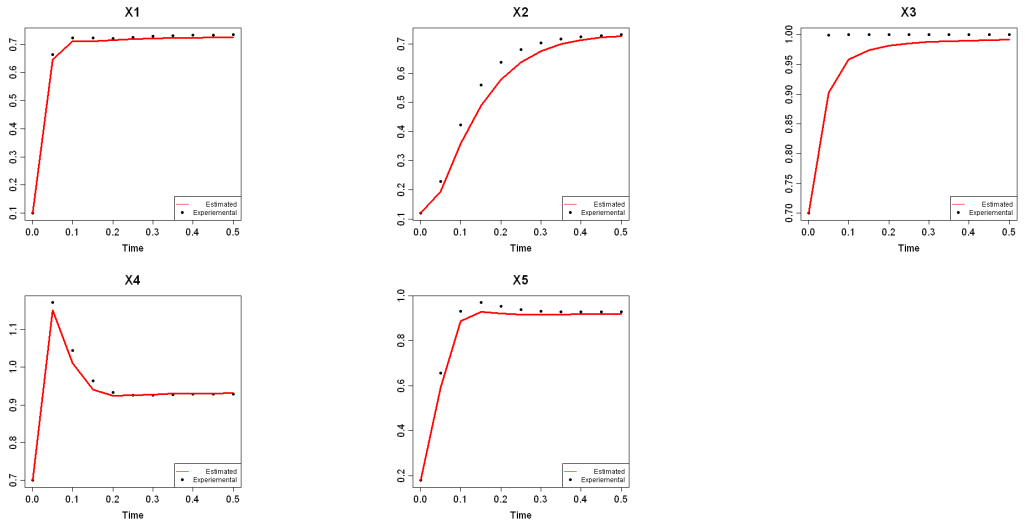


Figure 6.8: Comparison between expected behavior, i.e. solution of equation system in Figure 6.7 with parameters given in Table 6.4 (black circles) and estimated behavior obtained as a solution of the same equation system with the parameters inferred by KInfer (Table 6.5).

6.4 Real case studies

The analysis of in vivo time series for inferring the kinetic parameters is a worthwhile challenge. In this section we report the results obtained with KInfer for the estimate of the rate constants of three biologically relevant real case studies.

The first case study is the glycolysis and lactate production in bacterium L. lactis. The relative simplicity of the L. lactis metabolism, that converts sugars via the Embden-

Meyerhalf-Parnas pathway to pyruvate [192], makes the metabolic machinery of this bacterium an attractive case study for testing systemic approaches to modelling biochemical networks.

The second case study is the sub-network involving the IκB phosphorylation in the NF-κB pathway. NF-κB is a collective name for the complexes formed by the multigene family which functions as DNA-binding proteins and transcription factors. They are regulators of gene expression in eukaryotic cells but are held in an inactive state by a family of inhibitors (IκB). The biological relevance of this case study is due to the crucial role that NF-κB plays as the central mediator of inflammation with roles in cell death; and has been implicated in a myriad of common diseases - such as cancer, arthritis, asthma, diabetes, atherosclerosis and septic shock, to name but a few - and in the regulation of immune responses to infection. Recent detailed theoretical studies and experimental data about the NF-κB pathway can be found in [97, 98, 99, 138].

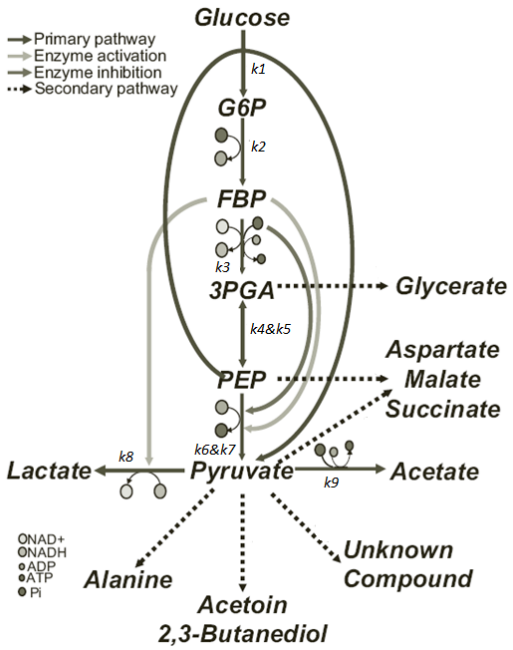
The third case study is the estimation of the kinetic constants of a gene transcriptional control mechanism in the cell cycle biochemical network of budding yeast. We modified a sub-part of a model in order to be able to include the CycB-triggered transcription of the Cdc20 gene. Then we used gene expression data to estimate some parameters that drive the periodic oscillations of chemical species connected to it.

6.4.1 Glucose metabolisms of Lactococcus lactis

We applied our method to infer the rate constants of the biochemical pathway that converts glucose into lactate in the bacterium *L. lactis* [77, 192] (see Figure 6.9). The experimental data provided by Voit consist of the time series of glucose (X_1), glucose-6-phosphate G6P (X_2), total fructose 1,6-biphosphate FBP (X_3), 3-phosphoglycerate 3-PGA (X_4), phosphoenolpyruvate PEP (X_5), pyruvate (X_6), lactate (X_7), acetate (X_8), ATP and inorganic phosphate P_i. The mathematical model of this pathway has been formulated by Voit et al. [77, 192] as in the equation system in Figure 6.9.

Parameter	Estimated value	Parameter	Estimated value
k_1	0.388 ± 0.001	k_6	0.0538 ± 0.0011
k_2	10.35 ± 0.011	k_7	0.0050 ± 0.0008
k_3	1.300496 ± 0.000012	k_8	0.00824 ± 0.00007
k_4	89.16 ± 0.12	k_9	0.012458 ± 0.000006
k_5	87.41 ± 0.10		

Table 6.6: Estimates of the kinetic rate constants of the pathway of regulation of glycolysis in L. lactis.



$$
\begin{aligned}
\frac{dX_1}{dt} &= -k_1 X_1^{0.4} X_5^{0.81} \\
\frac{dX_2}{dt} &= k_1 X_1^{0.4} X_5^{0.81} - k_2 X_2^{0.74} \mathrm{ATP}^{0.4} \\
\frac{dX_3}{dt} &= k_2 X_2^{0.74} \mathrm{ATP}^{0.4} - k_3 X_3^{0.88} \mathrm{Pi}^{0.01} \\
\frac{dX_4}{dt} &= 2k_3 X_3^{0.88} \mathrm{Pi}^{0.01} + k_4 X_5^{0.43} - k_5 X_4^{0.32} \\
\frac{dX_5}{dt} &= -k_4 X_5^{0.43} + k_5 X_4^{0.32} - k_1 X_1^{0.4} X_5^{0.81} + \\
&\quad -k_6 X_5^{0.53} X_3^{1.33} \mathrm{Pi}^{-0.0001} - k_7 X_5^{2.3} \\
\frac{dX_6}{dt} &= k_1 X_1^{0.4} X_5^{0.81} + k_6 X_5^{0.53} X_3^{1.33} \mathrm{Pi}^{-0.0001} + \\
&\quad + k_7 X_5^{2.3} - k_8 X_6^{0.46} X_3^{1.04} - k_9 X_6^{1.0} \mathrm{Pi}^{0.46} \\
\frac{dX_7}{dt} &= k_8 X_6^{0.46} X_3^{1.04} \\
\frac{dX_8}{dt} &= k_9 X_6^{1.0} \mathrm{Pi}^{0.46}
\end{aligned}
$$

Figure 6.9: Pathway of glycolysis and lactate production in *L. lactis*. Black arrows: flow of material; grey arrows: enzyme activation and inhibition; dashed arrows indicate leakage of material into secondary pathways, that are not considered in the model presented in this work. $X_{[1..8]}$ variables names are assigned as follows: X_1 = Glucose, X_2 = G6P, X_3 = FBP, X_4 = 3PGA, X_5 = PEP, X_6 = Pyruvate, X_7 = Lactate and X_8 = Acetate. The figure has been adapted from [77].

The parameter inference in this model suffers from difficulties of technical nature due to the peculiarities of the data. In fact, the time series of 3-PGA and PEP dip down very quickly, then recover, overshoot and slowly degrades. Even if it is possible to model such dynamics with generalized mass action law, the search algorithm might find a set of parameters causing the time course to cross over into the negative domain. In this case, non-integer reaction orders force the integration of the equation system to produce results with imaginary values. Moreover, some variables approach to zero toward the end of the experiment. Due to the numerical inaccuracies of any integration software, these variables may become negative. Also in our specific case, the search algorithm selects a parameter combination such that the simulation of the generalized mass action model with those parameters does not identify a global fit of the experimental data over their entire time domain. namely, the integration with the XPPAUT software stops at $t \approx 6.4$ min. Therefore, to avoid termination of integration, we artificially stopped the simulation of the dynamics of the pathway at $t \approx 6.4$ min, and extrapolated with the Stineman interpolation algorithm the behavior from $t \approx 6.4$ min till $t \approx 40$ min. We maintained the values of the orders of reaction as in Figure 6.9 and we estimated the kinetic rate

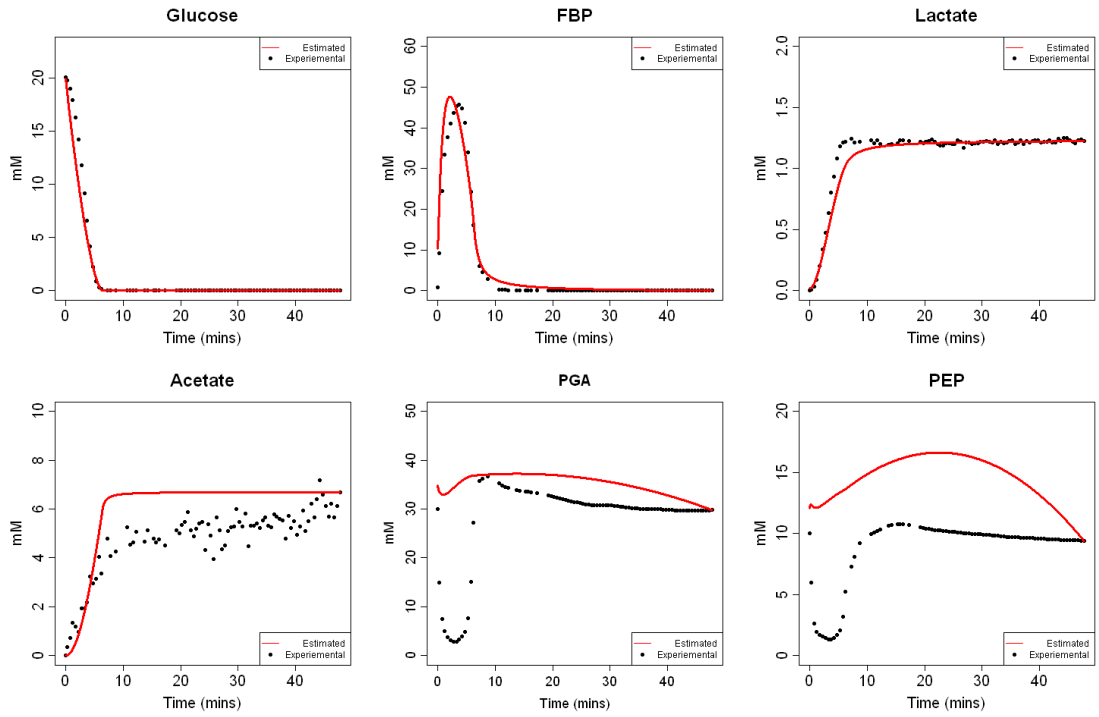


Figure 6.10: Comparison between experimental behavior (black circles) and estimated behavior obtained as a solution of equation system in Figure 6.9 with the parameters inferred by KInfer (Table 6.6).

constants of the model. Table 6.6 shows the KInfer estimates of these parameters. The rate constants estimates reproduce the experimental time series of the involved species, except for 3-PGA and PEP.

6.4.2 Binding affinity of IκB kinase

Activation of the NF-κB transcription factor can be triggered by exposing cells to a multitude of external stimuli such as tumour necrosis factor (TNF-α) and interleukin 1 (IL-1α). These cytokines initiate numerous and diverse intracellular signalling cascades, most of which activate the IKK complex. This IKK complex regulates the activity of the NF-κB transcription factor positively by phosphorylating the inhibitor - IκB. The IKK catalyses the transfer of the terminal phosphoryl group of ATP to the I-κB protein substrates, thereby tagging the inhibitor protein for ubiquitination and then degradation. The previously inactive NF-κB is thus activated and available for gene expression. This crucial component in the NF-κB activation cascade typically consists of two catalytic subunits, IKKα (IKK1) and IKKβ (IKK2), and a regulatory unit NEMO (IKKγ). The cytoplasmic inhibitors of NF-κB (the IκBs) are phosphorylated by activated IKK at specific N-terminal

residues, tagging them for poly-ubiquitination and rapid proteasomal degradation. Since recombinant human IKK2 (rhIKK2) phosphorylates GST-IκB in vitro, we examined the activity of the synthesised GST-IκB followed by the association and dissociation reaction, as shown in the chemical reaction system in Figure 6.11.

$$
\begin{aligned}
IKK2 + ATP &\xrightarrow{k_1} IKK2 \cdot ATP \\
IKK2 \cdot ATP &\xrightarrow{k_2} IKK2 + ATP \\
IKK2 \cdot ATP + GST\text{-}I\kappa B\alpha &\xrightarrow{k_3} IKK2 \cdot ATP \cdot GST\text{-}I\kappa B\alpha \\
IKK2 \cdot ATP \cdot GST\text{-}I\kappa B\alpha &\xrightarrow{k_4} IKK2 \cdot ATP + GST\text{-}I\kappa B\alpha \\
IKK2 \cdot ATP \cdot GST\text{-}I\kappa B\alpha &\xrightarrow{k_5} IKK2 + GST\text{-}I\kappa B\alpha\text{-}P + ADP \\
IKK2 \cdot ATP \cdot GST\text{-}I\kappa B\alpha &\xrightarrow{k_6} IKK2 \cdot GST\text{-}I\kappa B\alpha + ATP \\
IKK2 \cdot GST\text{-}I\kappa B\alpha + ATP &\xrightarrow{k_7} IKK2 \cdot ATP \cdot GST\text{-}I\kappa B\alpha \\
IKK2 \cdot GST\text{-}I\kappa B\alpha + ATP &\xrightarrow{k_8} IKK2 + ATP + GST - I\kappa B\alpha \\
IKK2 + ATP + GST\text{-}I\kappa B\alpha &\xrightarrow{k_9} IKK2 \cdot GST\text{-}I\kappa B\alpha + ATP
\end{aligned}
$$

Figure 6.11: The set of reaction modelling the activity of GST-IκB, in its associations and dissociations reaction with IKK2 [99]

A time course plot of the increasing of GST-IκBα-P was recorded to monitor the reaction advance (filled circles in Figure 6.13). The measured initial concentrations of the system's components are as follows: $[IKK2]_0 = 50$ nM, $[ATP]_0 = 200$ nM, $[GST\text{-}I\kappa B\alpha] = 1000$ nM, and $[IKK2 \cdot ATP]_0 = [IKK2 \cdot ATP + GST\text{-}I\kappa B\alpha]_0 = [GST\text{-}I\kappa B\alpha\text{-}P]_0 = [ADP]_0 \approx 0$ nM.

The only experimental time course measured for this model is the one of GST-IκBα-P. Nevertheless, from the law of mass action and assuming that ATP is in excess and thus its concentration does not change significantly in time, the time course of IKK2 and GST-IκBα can be derived respectively as: $[IKK2] \approx [IKK2]_0 - \frac{1}{k_{cat}}[GST\text{-}I\kappa B\alpha\text{-}P]'$ and

$$[GST\text{-}I\kappa B\alpha] \approx [GST\text{-}I\kappa B\alpha]_0 - \frac{c_2[GST\text{-}I\kappa B\alpha\text{-}P]'}{c_1 - c_3[GST\text{-}I\kappa B\alpha\text{-}P]'}$$

where the second equation is the best fit of the experimental data reported in Figure 6.12 ([99]), and $[GST\text{-}I\kappa B\alpha\text{-}P]'$ is the time derivative of the $[GST\text{-}I\kappa B\alpha\text{-}P]$.

The coefficients of the fit are: $c_2 = 2996.03$ nM min, $c_3 = 0.67078$ min. The inferred values of the rate constants are listed in Table 6.7. Figure 6.13 shows a good agreement within the error range, between the estimated time course and the measured time course of $[GST\text{-}I\kappa B\alpha\text{-}P]$.

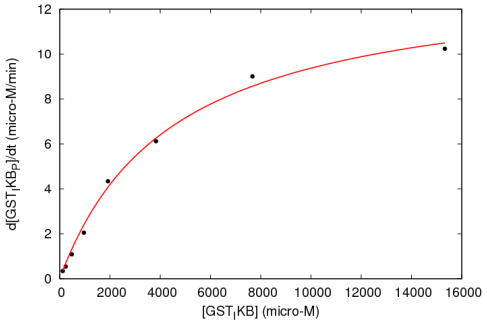


Figure 6.12: Reaction velocity (d[GST-IκBα-P]/dt) versus substrate concentration ([GST-IκBα]).

Parameter	Value	Parameter	Value
k_1	1.2308 ± 0.0008	k_6	0.043230 ± 0.000006
k_2	0.93060 ± 0.00018	k_7	12.247 ± 0.005
k_3	2.8506 ± 0.0003	k_8	1.351 ± 0.004
k_4	0.148654 ± 0.000012	k_9	0.4345 ± 0.0004
k_5	0.692538 ± 0.000018	σ	0.8006856

Table 6.7: Estimates of the model parameters. Δk is the experimental error propagating from the measures of concentration to the parameters, and σ_k is the variance of the parameter estimate; σ is the variance of the experimental uncertainty on the concentration measurements. The estimated values are in agreement with the experimental ones that can be found in [99].

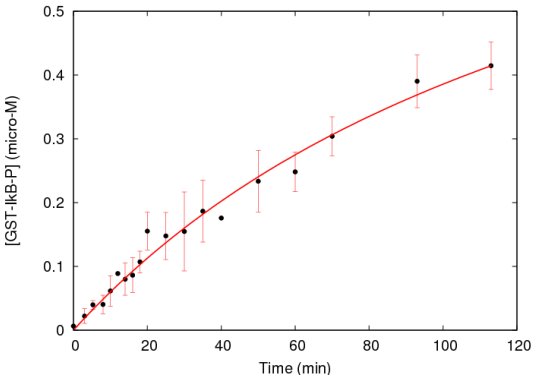


Figure 6.13: Time series of GST-IκBα-P. The error bars show the standard deviation associated with each measurements.

6.4.3 Transcription control of Cdc20 gene in a Budding Yeast cell cycle model

The last case study that we want to present is the estimation of the kinetic constants of a gene transcriptional control mechanism in the cell cycle biochemical network of buddying yeast. As reported in [1], the oscillations of the expression level of Cdc20 are very likely to be controlled by the cyclin-Cdk-dependent phosphorylation of gene regulatory proteins. In order to keep the model as simple as possible and to capture the essential feature of the triggering mechanism, we describe the regulation of gene Cdc20 transcription by CycB with the biochemical network depicted in Figure 6.14.

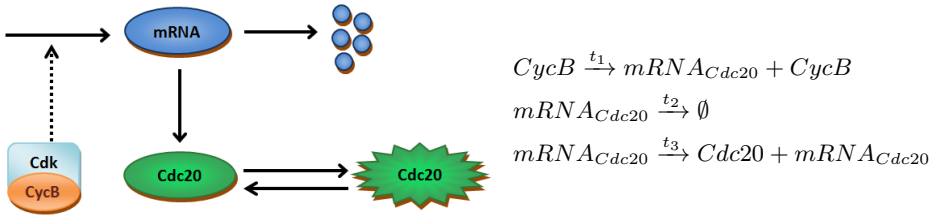


Figure 6.14: The transcription of the gene Cdc20 gene is regulated by the oscillations of Cdk/CycB. The regulation depicted in the cartoon corresponds to the set of reactions on the right.

We modified the original Novak and Tyson model presented in [142] by including the CycB-triggered transcription of the Cdc20 gene. We introduce this modification in order to be able to apply our inferring strategy on the experimental data in Cyclebase database [72] for the expression level of the gene coding for this specific protein of the cell cycle control. In Figure 6.15 the curves of the expression level of Cdc20 gene obtained in six experiments are shown. Table 6.8 summarizes the main information contained in the experimental data [72].

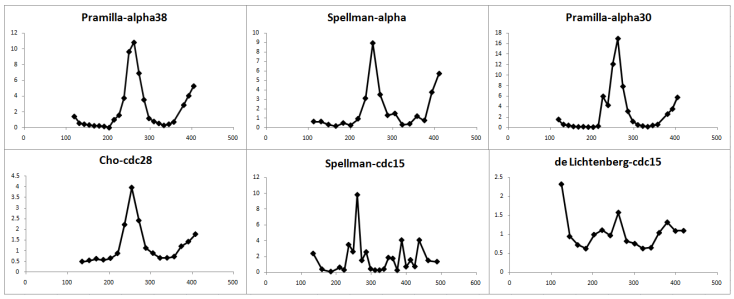


Figure 6.15: Six experiments denominated Pramilla-alpha38, Spellman-alpha, Pramilla-alpha30, Cho-cdc28, Spellman-cdc25, and de Lichtenberg-cdc15 provide the time series of the expression level of Cdc20 gene during one cell cycle [72, 175, 51].

Experiment	**Rank**	P_{per}	P_{reg}	**Peaktime**
Pramilla-alpha38	50	0.000025	0.0274	77%
Spellman-alpha	182	0.0068	0.0512	74%
Pramilla-alpha30	46	0.00001	0.0203	75%
Cho-cdc28	439	0.0042	0.6365	uncertain
Spellman-cdc15	303	0.0509	0.0194	uncertain
de Lichtenberg-cdc15	183	0.0097	1.237	76%

Table 6.8: Rank, P-values and peaktime of the six experimental micro-array time series shown in Figure 6.15. The *rank* orders each gene of an organism by a score that has been assigned on the basis of its pattern of expression and magnitude of regulation. Those genes with the highest periodicity and that are most regulated are given the best ranks (lower number). The P-value for periodicity (P_{per}) is the chance of observing as great a periodicity by random shuffling of the individual time point values of the expression profile. A small P_{per} value therefore implies a highly periodic pattern of expression. The P-value for regulation (P_{reg}) estimates the chance that the magnitude of regulation will have occurred by chance. A small P_{reg} values therefore implies a strongly regulated gene. The *peaktime* describes when in the cell cycle a gene is maximally expressed. Peaktime is represented as a percentage with both 0 and 100 representing the M/G1 transition in the cell cycle. In some cases the peaktime will be marked as uncertain. There are several reasons this uncertainty can occur: the experiment(s) are not sufficiently periodic for a peaktime to be determined or more than one experiment has been analyzed but the experiments disagree with respect to the time of peak expression. In such cases, inconsistency across the different experiments makes it impossible to calculate a reliable peaktime.

The modification on the structure of the model that we introduced is the definition of the dynamics of CycB regulation on the expression level of Cdc20 gene (i.e. $[mRNA_{Cdc20}]$) with the following rate equations.

$$\frac{d[mRNA_{Cdc20}]}{dt} = t_1 \cdot [CycB] - t_2 \cdot [mRNA_{Cdc20}] \tag{6.1}$$

$$\frac{d[Cdc20]_T}{dt} = t_3 \cdot [mRNA_{Cdc20}] - k_6 [Cdc20]_T \tag{6.2}$$

In spite of the fact that Eq. (6.1) and (6.2) look like minor changes of the original model, we will see that the time series of all the species involved in the model are highly sensitive to the parameters t_1, t_2 and t_3.

The parameters t_1, t_2, and t_3 introduced in Eq. (6.1) and Eq. (6.2) can all be inferred from the experimental dynamics of the expression levels of Cdc20 gene (Figure 6.15) and from the experimental measurements of the time series of $[CycB]$, since the inference model implemented by KInfer takes as an input the experimental time course of all the reactants. Note that, time resolved measurements of $[CycB]$ are not available, since, with the current techniques, they are quite hard to obtain experimentally. However, as we show in Figure 6.14, in our model the mechanism with which CycB regulates the synthesis of the mRNA of Cdc20 gene, is abstracted using the following virtual reaction: $CycB \longrightarrow$

$mRNA_{Cdc20} + CycB$. Assuming, that the $[mRNA_{Cdc20}]$ oscillations are almost entirely driven by the oscillation of $[CycB]$, and, to a lesser extent, by the mRNA degradation, we have the following inequality $|d[mRNA_{Cdc20}]/dt| \gg t_2[mRNA_{Cdc20}]$, $\forall t$, that allows us to obtain the approximation $d[mRNA_{Cdc20}]/dt \approx t_1[CycB]$, with $t_1 \in (0, 1]$. Therefore, the missing experimental time course of $[CycB]$ can be obtained from $d[mRNA_{Cdc20}]/dt$, i.e. slope of the tangent to the experimental curve of $[mRNA_{Cdc20}]$ in the measured time points.

The inference procedure applied to different experimental datasets gives different sets of parameter estimates, that we report in Table 6.9, where each parameter estimate t_p is associated with its error Δt_p ($p = 1, 2, 3$), due to the propagation of the experimental uncertainty from the input concentration time series (see Section 5.1.2).

Experiment	$\mathbf{t_1 \pm \Delta_{t_1}}$ ($\mathbf{min^{-1}}$)	$\mathbf{t_2 \pm \Delta_{t_2}}$ ($\mathbf{min^{-1}}$)	$\mathbf{t_3 \pm \Delta_{t_3}}$ ($\mathbf{min^{-1}}$)
Spellman-alpha	**0.122 ± 0.002**	**0.0116 ± 0.000835**	**0.0129 ± 0.0045**
Spellman-cdc15	0.126 ± 0.0007	0.0107 ± 0.00002	0.0211 ± 0.00603
Cho-cdc28	0.235 ± 0.0198	0.0395 ± 0.00869	0.0127 ± 0.0035422
Pramilla-alpha30	**0.0857 ± 0.00144**	**0.0146 ± 0.0016**	**0.019 ± 0.005773**
Pramilla-alpha38	0.14 ± 0.0001	0.0216 ± 0.00022	0.0193 ± 0.00546
de Lichtenberg-cdc15	1.884 ± 0.5308	0.0995 ± 0.03583	0.0125 ± 0.003617

Table 6.9: Parameter estimates of the Cdc20 transcription sub-network. The experiments from which the inferred kinetic constants do not reproduce the expected oscillatory behavior of the model's proteins are highlighted in bold.

Experiment	$\sigma_{\mathbf{t_1}}$ ($\mathbf{min^{-1}}$)	$\sigma_{\mathbf{t_3}}$ ($\mathbf{min^{-1}}$)	$\sigma_{\mathbf{t_3}}$ ($\mathbf{min^{-1}}$)
Spellman-alpha [175, 72]	0.33	0.0021	4.46×10^{-4}
Spellman-cdc25 [175, 72]	0.034	0.0027	0.0025
Cho-cdc28 [72]	0.086	0.0028	4.18×10^{-4}
Pramilla-alpha30 [72]	0.021	0.0061	0.00266
Pramilla-alpha38 [72]	0.034	0.0054	0.0025
de Lichtenberg-cdc15 [51, 72]	0.232	0.006	0.0014

Table 6.10: Variances of the estimated parameters (σ_{t_1}, σ_{t_2}, σ_{t_3}).

Each set of parameters (t_1, t_2, t_3) defines a new model. The solutions of the six new models are shown in Figure 6.17. The parameters inferred from "Spellman-alpha" and "Pramilla-alpha30" experiments do not reproduce the expected oscillatory behavior for any of the species included in the model. As we can see in Table 6.10, the variances

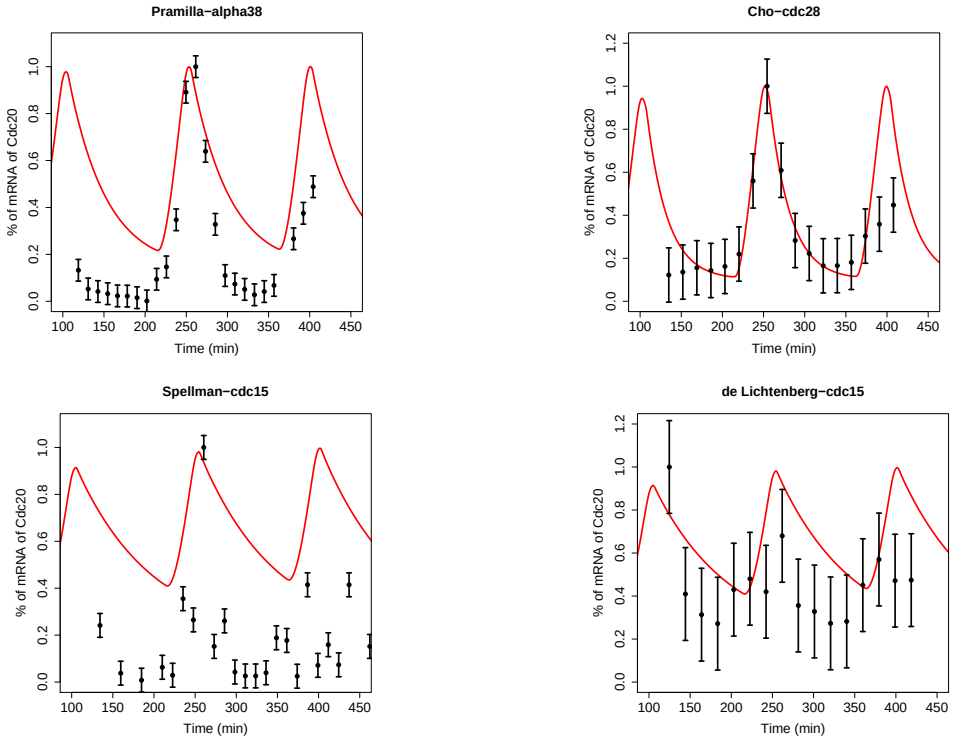


Figure 6.16: Comparison between experimental time course of $[mRNA_{Cdc20}]$ and the simulated time course with the parameters t_1, t_2, and t_3 inferred from the four experimental time series in Figure 6.15. Only the four datasets which mantain an oscillatory behaviour are shown.

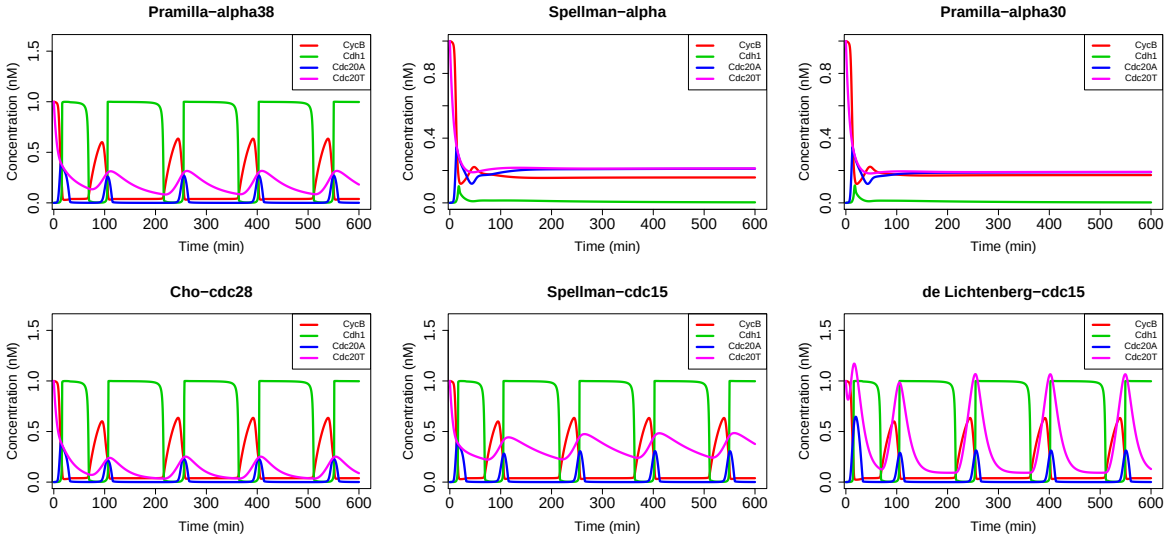


Figure 6.17: Solutions of the model including Cdc20 transcription process with the rate coefficients in Table 6.9 inferred from the six experimental datasets shown in Figure 6.15.

of parameters t_1 and t_2 in these two cases are one order of magnitude bigger than the value of the corresponding parameter estimates, revealing a big inaccuracy and a low confidence in the estimates of these parameters. This results also reveals that the cell cycle model considered in this study is sensitive to t_1 and t_2. In Table 6.8 we notice that the experimental data of the expression level of Cdc20 in Spellman-alpha experiments have a high value of P_{per} indicating a low periodicity in the expression pattern. This fact may justify the high variances of the parameter estimates inferred from this experiment. However, the P_{per} value of the Spellman-alpha time series is not the highest and it has the same order of magnitude of the P_{per} obtained in the experiment Cho-cdc28 from which the inference procedure estimated parameters able to reproduce the expected oscillatory dynamics. The highest P_{per} values is reported in the experiment of Spellman-cdc15, that also reports a strong uncertainty on the peaktime. Nevertheless, the parameters inferred from Spellman-cdc15 still guarantee the oscillatory behavior of the species concentration (see Figure 6.17), even if Figure 6.16 shows in the case of this experiment the highest disagreement between the estimated and the experimental expression level of Cdc20 gene. Further analysis about the relationship between the variance associated with the estimated parameters and the values that are quantifying the periodicity of the data has been performed and can be found in [118].

We conclude this section by pointing out that the new model defined by the reactions in Figure 6.14 is simpler than the Hill mechanism used in the original model, but it is equally able to reproduce the observations. In a general case, however, we could still use the Hill kinetics and use KInfer to infer its parameters following the strategy introduced in Section 5.1.3 by approximating the Hill function with a Gauss error function as in Eq. (5.23), where the parameter estimates used for this example are reported in Table 6.11. In Figure 6.18 the goodness of the fit of the KInfer estimated `erf` function to the Hill form is showed. We decided to introduce the modification of the model structure just to be closer to the elementary mechanistic reactions driving the oscillations of the different mRNA/proteins.

Parameter	Value	Asymptotic standard error
c_1	10.36	0.1%
c_2	2	–
c_3	2	–
c_4	1.159	–
c_5	-10.16	0.1 %

Table 6.11: KInfer estimates of the parameters of the Gauss error function approximating the Hill function $h(x) = \theta \frac{mCycB^n}{c^n + mCycB^n} \approx c_1 \left[\texttt{erf}(c_2 \cdot mCycB^{c_3} + c_4) \right] + c_5)$. Parameters c_2 and c_3 are given as an input by the user. c_4 is equal to σ_{exp} throughout all the experiments (see Figure 6.15).

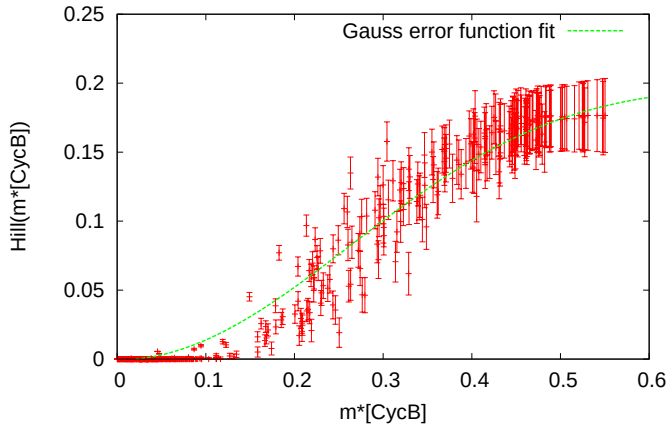


Figure 6.18: Fit of Gauss error function to Hill function. The width of the error bars have been calculated by propagating through the Hill equation an experimental error on the measurements of $[CycB]$ equal to the 7% of the measurements.

6.5 Other existing inference tools

For the parameter inference problem, many methods and tools have been proposed in the systems biology community. In Section 2.2 the main competing approaches have already been described. In this section we select some available software tools implementing the different representative ideas and we explain their application on some examples in order to stress the features that distinguish our approach from the other ones.

COPASI [96]. COPASI is a platform-independent software tool that offers several features for the simulation and analysis of biochemical reaction networks. For what concerns the parameter estimation problem, COPASI is equipped with a number of diverse optimization algorithms that can be used to minimize or maximize different variables of the model. A special case of optimization is parameter estimation. The objective function is given implicitly by a function that measures the distance between the model and the experimental data, such as a sum of squares of residuals. The kinetic constants of the model are adjusted to minimize the objective function. Different methods are implemented in the tool: two algorithms based on estimating derivatives of the objective function (steepest descent and Levenberg Marquardt), a direct search algorithm, and different kind of

evolutionary algorithms. The software is able to simultaneously fit the model to data from steady-state and time course experiments. This is achieved by enabling the user to provide multiple data files with multiple experiments.

The main advantages of this software platform are that it has a user friendly GUI that facilitates a modeler to easily switch between different simulation approaches and analysis/plotting techniques. Moreover it reads and writes SBML files and it supports the export of the model in C code and Berkeley Madonna file. The main disadvantage for what concerns the parameter estimation problem is that it implements a fitting procedure which can be inadequate in a context in which we suppose to have very noisy data and for which the probability distribution underlying the data needs to be considered.

SBML-PET [199]. SBML-PET is a Systems Biology Markup Language based Parameter Estimation Tool. This tool has been designed to do parameter estimation for biological models. It can run on Linux and Cygwin on Windows and, using a shell-like interface, allows the import of a SBML model together with experimental dataset possibly produced under different experimental conditions. These two inputs are used to build a cost function that is optimized using an Evolutionary Algorithm in order to get the best parameter fit. The tool uses an ODE Solver and the cost function is evaluated taking into consideration the number of experimental conditions, the number of time courses, the number of experimental data, the number of sampling points available and the prediction data from the mathematical model. The cost function also include weights that are calculated from the standard deviation associated with the experimental measures (if available).

The main advantage of this platform is the fact that it has been explicitly designed to support the standard language in which many existing model are currently written (SBML), but the not so easy user interface and the fact that it implements a standard fitting procedure for deterministic models, make it not so appealing for analyzing models of biological systems with noisy input data and where the stochastic effect should be taken into consideration.

PET [172]. PET (Parameter Estimation Toolkit) is a graphical user interface for exploring the parameter space of a mathematical model. PET is designed for biological models based on Ordinary Differential Equations using SBML as the modelling language. For the parameter estimation issue, in PET a weight is assigned to every experimental datum in order to indicate the relative importance of that datum compared to the other data. The optimization algorithm uses the weights when fitting the model to experimental data using as objective function the weighted sum of the squares of the orthogonal

distances between the experimental data and the model simulations. In [200] this tool has shown to be useful in the estimation of parameters of a deterministic model of cell cycle regulation in budding yeast.

The advantages/disadvantages of this tool are similar to the ones listed for the SBML-PET tool, apart from the issue about the usability of the interface that, in this case, is much easier because it is based on graphical objects like windows, buttons and textfields rather than just shell commands like the SBML-PET tool.

BioBayes [193]. BioBayes is a software tool that supports standard definitions of mathematical models and provides a framework for applying methods of Bayesian inference to ODE models of biochemical systems. Through an user friendly graphical user interface, users can load models and define the desired prior distributions for the kinetic parameters and run different samplers (i.e. Metropolis-Hastings, population-based MCMC) to infer their posteriors using one or more experimental datasets.

Some of the advatages of this tool (that are features of the Bayesian approach in itself) are that, due to its probabilistic nature, it allows one to consider noisy observations as a source of data for learning full distributions of beliefs and it provides a formal way in which prior knowledge can be included in the modelling process. The main disadvantages are that the convergence to a good estimate of the parameter value is usually not guaranteed and it depends both on the data used for the inference and on the priors chosen for running the estimation procedure, even if the possibility of using non-informative priors [183, 48] can also be employed (the idea behind their usage is to make inferences when external information is not available. The uniform distribution is frequently used as a non-informative prior).

Chapter 7

Discussions and Future work

In this thesis, we presented a work that is part of a larger, multidisciplinary project involving many researchers for the development of conceptual and computational tools for modelling, analysis and simulation of multi-level/multi-scale biological systems to predict their behavior in a modular, compositional, scalable and executable manner. The aim of those tools is to become an artificial laboratory in which it is possible to replicate in-silico the activities that are usually performed in real wet labs.

Within a project as large and varied as this one, there are numerous areas where new applications of the work presented in this thesis can be applied and incremental improvements of both the methods and the tools can be made. In the rest of this chapter we try to sketch the possible future development that we feel to be the more appealing.

Extensions to the inference framework

The first and more natural incremental improvements of the inference framework presented in this thesis, are some additions that can be easily integrated in the actual implementation of the KInfer tool and that are useful in order to be able to handle in a better way more complex case studies in which not all the information are known to the user.

An important remark that needs to be made, indeed, is that the time-resolved experimental data suitable for an accurate maximum likelihood inference of parameters should possibly satisfy two main requirements: 1) an optimal sampling time selection; 2) the time series of the majority of the species involved in the system should be available. However, the experiments to generate data are complex and expensive, as a consequence of which the time series available are usually rather short, with few (if any) replicates. Morever, almost certainly, not all the variables that the user would like to include in the model can be measured. In the light of this, an important feature to be added to the current version of KInfer are modules performing the following tasks: 1) find the optimal sampling time

that reduce the variance of the error in the estimated parameters (this task is of practical importance in experimental design); 2) simultaneous inference of parameters and missing time-series through the optimization of the same likelihood function of observed data in which missing time series are treated as missing parameters.

Other improvements that can make KInfer easier to use are the implementation of modules able to combine different time series replicas in the same model, in order to infer a set of parameters that matches more than one single experiment and the implementation of modules able to parse and include in the likelihood function rate laws different from the general mass-action (e.g. Hill responses, algebraic equations accounting for conservation laws). The implementation of these modules is a fundamental improvement needed in order to apply KInfer in wider contexts.

In a more general view, interesting future developments of the inference framework are the integration of the parameter estimation step with the inference of the structure of the network. The investigation of some correlation-based approaches for network inference has already been started, but at the time of writing the results are still not ready to be integrated in the framework [117].

Moreover also the identifiability problem should be addressed [164]. Parameter identifiability is a fundamental prerequisite for model identification; it concerns uniqueness of the model parameters determined from the input-output data, under ideal conditions of noise-free observations and error-free model structure. Recently, differential algebra methods seem to be promising tools to study the identifiability of nonlinear dynamic systems described by polynomial equations [166, 123]. Given that biological/physiological systems are usually characterized by nonlinear dynamics, e.g. threshold processes, and that the identification experiments are often performed on systems started from known (equilibrium) initial conditions, a future step of the work presented in this thesis is to develop new differential algebra algorithm, which tests a priori identifiability of nonlinear models with given initial conditions.

Finally, some considerations need to be made about the integration of the two frameworks (i.e. the modelling and the inference one) presented in this thesis. Although the two can be used independently and in isolation to accomplish interesting analyses like the ones presented in the previous chapters, having an integrated environment in which the two software systems can be linked would be highly desiderable.

Since the model's formalism that KInfer uses are equations in the form of general mass action, the first natural way to approach the integration between the two frameworks is to

try to automatically deduce this set of equations from the BlenX specification of a system. The derivation of a system of ordinary differential equation from a process algebra model (often also referred as fluid-flow analysis) has been studied by several authors. Between the main contributions we cite the work done by Hillston and colleagues [92, 13] on different classes of PEPA models; a fluid-flow formulation for the stochastic constraint programming language (sCCP) by Bortolussi and colleagues [12]; various translations, presented by Cardelli [22, 23, 21], from stochastic process algebras like stochastic π-calculus to systems of chemical reactions and back. In all the aforementioned studies a specific subclass of the language and syntactical restrictions have been introduced in order to derive ODEs by static inspection of the model description. Some following studies propose methods to overcome those limitations [88, 185] and prove the convergence to the same behavior of the two formalisms; other studies also deepening the analysis of the fundamental characteristics of the derived ODEs, such as the existence, uniqueness, boundedness and nonnegativeness of the solution [58]. Sometimes semantics different from the original one needed to be introduced in order to achieve those goals.

Many peculiar features of BlenX make it more complex with respect to the other process algebra mentioned above for which some study of their relationship with the ODEs has already been carried out. Hence, any automatic derivation of ODEs from a BlenX model will need to take inspiration from the works cited above but it will for sure need to address more difficult theoretical issues. The main topics that need to be very carefully defined concern the limitation on the kind of actions/events contained in the model and methods of identifying the generation of a possible infinite state space. The study of those subjects represent interesting development of the theory behind the BlenX language: the successful characterization and implementation of those topics will allow the final user to be able to handle in a unique framework deterministic and stochastic approaches on a specific model of choice and/or apply in a transparent way the parameter inference procedure described in this thesis.

Biological systems in a stochastic framework

The study of the budding yeast cell cycle allowed us to identify many interesting analyses that can be further carried on in order to better understand the behavior of this organism and that can be useful also for studying properties of other biological systems.

For example, a systematic quantification and analysis of the partially viable mutants already experimentally characterized can give us information about the completeness and reliability of the model in the stochastic framework. Inconsistency between synthetic and measured data can point to parts of the model that need to be further refined, whereas a

good agreement between the two will open the possibility of using the model to test new hypotheses and answer new questions.

Moreover, a deeper investigation of the "intrinsic vs. extrinsic" noise affecting the budding yeast cell cycle is a natural evolution of the studies carried out in this thesis. More detailed analyses like the ones in [101, 8] on refined models can reveal where and from which sub-module of a model the noise can arise. This is important in order to understand if the intrinsic fluctuations of molecular numbers is the sole cause for the hetereogeneity of cell cycle measures (e.g. size at division, cell cycle time, etc.) or only some kind of extrinsic noise (derived from a part of the system not included in the model) can make the in silico results in agreement with the wet-lab data.

The kind of analyses described above are useful for any kind of biological system (oscillatory and not): however many of the results presented in this thesis have been obtained using ad-hoc pieces of code adapted in order, for example, to analyze BlenX output traces and get the frequency/amplitude of specific oscillating species. Also the pedigree study has been tuned on that specific model of cell cycle, containing peculiar species with an associated precise interpretation, e.g. the threshold's traversal of a specific species define the transition between different phases of the cycle. A more general framework able to handle different kind of models, where the modeller can specify (possibly in a user-friendly way) which are the characteristics of the model he/she is interested in measuring during the simulation can be a useful and interesting extention of the BlenX simulation framework.

The way in which, in recent years, biological systems has been studied looks strikingly similar to the approach of analyzing electronic circuits: in the wiring diagrams modelling biological processes, instead of resistors, capacitors and transistors hooked together by wires, one sees genes, proteins and metabolites connected by chemical reactions and intermolecular interactions. The natural consequence is to ask whether physiological regulatory systems can be understood in modular terms, in the same way an electrical engineer would model a radio [114].

Being able to isolate, characterize and analyze the single modules that compose a complex behavior will be an interesting addition to the study of the budding yeast cell cycle, especially in the stochastic framework. The advantages of isolating modules contained in the cell cycle model, like hyperbolic/sigmoidal responses, positive/negative feedback loops or irreversible/hysteretic switches are two-fold: from one side, the modeller can create a library with all those behaviors that later can be used to compose new models in an easy way; from the other side each module can be studied in isolation in order to

understand the effect of the noise on the system's behavior [17]. Having different versions of the same module (i.e. having the same overall modelled response, but, for example, with modules that in isolation are subject of a different coefficient of variation), the modeller can study the effect of each single version of the module in the context of the whole model and can deduce which are the most sensitive part of the model with respect to the noise (i.e. if the experimental measurements show a less noisy response compared to a module modelled with all its elementary steps it means that the real system possess a not yet considered external mechanism able to reduce the overall effect of the noise).

Finally approaches similar to the semi-automatic translation of ODEs to BlenX presented in this work should be taken into consideration in the overall idea of developing a framework in which also non-expert computer scientists can build models of biological system in BlenX. Allowing the user to import a model written in other formalisms/languages (possibly more familiar to people coming from a biological background) will increase the impact that process calculi approaches and related tools/techniques can have on the analysis of biological systems. Indeed only if experts in the system under consideration (and not experts in the chosen modelling approach) develop a specific model, then meaningful insights on the biological behavior can be obtained quickly and reliably: so an effort in blurring the boundaries between the expertises needed to think about what the user wants to model and the actual writing of the model in BlenX is a worthwile field of investigation and efforts in this sense have already been started.

Chapter 8

Conclusions

In this thesis we presented a contribution to the model building cycle for systems biology.

In particular, we showed that the use of a process algebra approach for creating models of biological systems can allow the study of properties that are not easily accessible in the classical formalism used for modelling biological systems. The literature is rich in "ready-to-use" complex models which have been analyzed in detail and characterized, and the knowledge contained in those validated model needs to be transferred to models written with different languages as easily as possible in order to allow the discovery of further incremental insights about the system. With this idea in mind, we presented a translation of ordinary differential equations into BlenX, a process algebra language specifically designed to model networks of interacting chemical species. We showed on the concrete case study of budding yeast cell cycle, that the translation is straightforward and, thanks to the peculiar features of the BlenX language, can mantain the same level of abstraction as the original models so that the knowledge contained in those already validated models can be easily transferred and compared to the results obtained in the stochastic framework.

The relevance of studying this specific biological system in the stochastic BlenX framework can be deduced from the results that our approach allow us to find. In particular we were able to investigate models of the budding yeast cell cycle of increasing levels of detail and complexity: we quantified the strength of the irreversibility of the "Start" transition of the cell cycle with respect to a specific parameter in a simple core model. This kind of result allows the user to get a quantitative answer to the question of which parameters are the ones that affect key properties of a system. Following the idea of studying systems of increasing levels of complexity, we analyzed a more detailed model of the budding yeast cell cycle with respect to some peculiar mutants at the border of life and death for which the noise can play a crucial role for their survival. We found that the stochastic characterization of those systems was in a greater accordance with the experimental data

available. Finally, inspired by the kind of single-cell experiments that are carried out in a wet-lab for studying budding yeast, we implemented a framework for performing in-silico pedigree analysis of the asymmetrical growing and division of budding yeast models. In particular we chose the most complete model of the system available in the literature and for which complete and detailed both experimental and theoretical studies have been published. We showed that all the simulation results that we obtained for different kind of analyses (e.g. checking the synchronicity of different colonies, partial viability seen as the generation of an incomplete pedigree tree, ...) have been validated against the experimental evidence of the biological system. The nice agreement between the in-silico and experimental results make us confident that this framework, can be used for gaining more deep insights in the study of the budding yeast growing and division process.

Modelling approaches are central in systems biology, as they provide a rational framework to guide systematic strategies for key issues in medicine as well as the pharmaceutical and biotechnological industries. The estimation of parameter values (model calibration) is the bottleneck of the computational analysis of biological systems. Inter- and intra-cellular processes require dynamic models, that contain the rate constants of the biochemical reactions. These kinetic parameters are often not accessible directly through experiments, therefore methods that estimate rate constants with the maximum precision and accuracy are needed. We presented a new method for estimating rate coefficients from noisy observations of concentration levels at discrete time points. This is traditionally done by fitting procedures (e.g. by the least-squares estimator). We propose an alternative approach based on a probabilistic, generative model of the variations in reactant concentration. Our method, starting from the time series concentrations for the $X_1, \ldots, X_N$ chemical species, discretizes the law of mass action, and as a result provides a tool to predict the values of the variables X_i at time t, based on their values at the previous time point. The discretization of the law of mass action provides a model for the variations of the species concentration, rather than a model for the time-trajectory of the species concentrations. This makes the evaluation of the expectation value of the mass action law function simpler and analytically tractable. The probabilistic formulation of the method is key to a principled handling of the noise inherent in biological data. The method also includes a simple strategy to automatically estimate the initial guesses and bounds for the parameters: in this way the user, especially if he/she has no idea about their values, is not asked to give them as input. We implemented this mathematical procedure into a software tool, called KInfer. The tool has been shown to be able to obtain estimates for the rate coefficients in case studies of different complexity including first and second order chemical reactions, didactical examples of biochemical networks, and more com-

plex biological pathways like cell cycle regulation mechanisms and the NF-kB pathway. The results obtained on these case studies confirmed that the procedure converges to the expected solution within the noise strength and experimental error on input data.

Summing up, the contribution of this thesis can be seen as an effort to apply approaches and tools born in the computer science field to the modelling and the study of biological systems, in a way that could lead, one day, to a more easy and automated way to analyze biological case studies.

Acknowledgements – Ringraziamenti

E anche il momento dei ringraziamenti è arrivato... Tre anni sono passati e il bilancio da parte mia è più che positivo e questo solo grazie a tutte le persone che qui ringrazierò senza fare nomi, senza alcun ordine di priorità/precedenza/importanza... perchè tutte le persone che in un modo o nell'altro hanno incrociato le loro strade con la mia mi hanno dato qualcosa e questo qualcosa ha fatto di me quella che sono oggi.

Quindi GRAZIE:

... a chi ha rallegrato le mie giornate con canzoni alle volte alquanto inquietanti, ma sempre apprezzate per lo spirito con cui erano eseguite (e anche per la qualità vocale, non volevo insinuare il contrario ;-))...

... a chi ha scelto di rendermi partecipe delle sue vicende e stress quotidiani e si fida dei miei consigli nonostante la mia visione cinica del mondo...

... a chi ha tentato di farmi diventare un po' più eco-pollice-verde (senza tanto successo)...

... a chi non si lamenterebbe neanche se ricoprissi l'intero soffitto della camera con le mie foto (e sappiamo che c'è mancato poco :-))...

... a chi, al ritorno da qualunque missione (segreta e non), aveva sempre un pensiero azzeccatissimo per me...

... a chi aveva acceso un piacevole sottofondo musicale nella mia vita ma poi ha deciso di spegnerlo...

... a chi è ancora giovane ma sta con grande determinazione trovando la sua strada per la realizzazione personale...

... a chi non mi crede quando dico che gli asparagi non mi piacciono e non è una scusa! V.v.V ...

... a chi ha condiviso aspetti paralleli della mia vita e che mi ha fatto giungere alla conclusione che per me è difficile essere multi-core...

... a chi mi ha dimostrato che ci sono diversi gradi di tirannia e che alcuni tiranni sono delle persone su cui si può fare sempre affidamento...

... a chi ha condiviso situazioni di frustrazione professionale nel percorso di dottorato e che è un'altra testimonianza del motto "Success is the best revenge", indeed :-) ...

After all you put me through, You'd think I'd despise you, But in the end I wanna THANK YOU, 'Cause you made me that much stronger [...]

[...] Cause if it wasn't for all that you tried to do, I wouldn't know, Just how capable I am to pull through, So I wanna say thank you, Cause it, Makes me that much stronger, Makes me work a little bit harder, It makes me that much wiser [...]

... a chi ha trovato un senso, "anche se questa vita un senso proprio non ce l'ha"...
... a chi, anche se non viviamo più sotto lo stesso tetto, non si dimentica della sua mogliettina e mi fa sempre sentire il suo supporto anche da lontano ...
... a chi sta cominciando ora a capire che stare fuori dall'Italia ha i suoi pro e i suoi contro (pro = solitamente bel tempo atmosferico, contro = solitamente poco tempo rimasto dopo gli intensi esami/report/papers/presentazioni che ti danno da fare)...
... a chi, anche se presa da mille e mille attività, rimane sempre il mio girasole preferito...
... to wonderful people that showed me what Scottish friendliness and kindliness really mean (both personally and professionally)...

Essendo questo il risultato di un lavoro di dottorato, ovviamente, ringrazio il mio supervisor (Corrado Priami) e tutti quelli che hanno lavorato con me (prima tra tutti Paola Lecca): li ringrazio per avermi dato la possibilità di confrontarmi con la realtà della ricerca e per avermi permesso di incontrare tante persone nel mondo che fanno questo lavoro, perchè le esperienze a Los Angeles, Seattle, Valencia, Edinburgh, Cambridge (già, Cambridge...), Honolulu, Palma de Mallorca, etc. mi hanno veramente aiutato a crescere, sia professionalmente che personalmente.

A very special thank goes to a great person that supported me (e anche sopportato) in these three years and during the last tough period of my PhD: thanks to your wise comments, corrections and opinions, Attila, I think that finally, after few downs and thanks to a wonderful pass from you, I was able to score a great touchdown!! :-D and you know what I'm talking about ;-)

And my true gratitude goes to my reviewers (Prof. Hillston and Prof. Tyson) that provided me precious critiques, comments and corrections on my work: the helpful discussions that I had with both of you really improved the overall quality of this thesis... but any error, incomprehensible sentence or incoherence in the logic of the topic described is just my own fault!!

Infine, come dicono gli inglesi, last but not least, ringrazio tutta la mia famiglia (in particolare la Mutti, il Papà e il Brother) che in tutti questi anni mi ha sostenuto emotivamente, concretamente e anche economicamente... anche se qualche volta il fatto che nella ricerca non sappiamo bene cosa stiamo cercando fino a quando non l'abbiamo trovato è un po' difficile da conciliare con la domanda "ma cosa devo rispondere a chi mi chiede –che cosa studia tua figlia?–"... del resto qualcuno ha detto "If we knew what it was we were doing, it would not be called research, would it?" (cioè, se avessimo saputo cosa stavamo facendo non si sarebbe chiamata ricerca, no?)... l'ha detto un certo A. Einstein...

[...] So thanks for making me a fighter, Made me learn a little bit faster, Made my skin a little bit thicker, Made me that much smarter, Cause if it wasn't for all of your torture, I wouldn't know how to be this way now and never back down, So I wanna say THANK YOU (C. Aguilera - Fighter)

Bibliography

[1] B. Alberts. *Essential cell biology: an introduction to the molecular biology of the cell*. Garland, 1998.

[2] N. A. Allen, K. C. Chen, J. J. Tyson, C. A. Shaffer, and L. T. Watson. Computer evaluation of network dynamics models with application to cell cycle control in budding yeast. *IEE System Biology*, 153:1321, 2006.

[3] A. P. Arkin and C. V. Rao. Stochastic chemical kinetics and the quasi-steady-state assumption: application to the Gillespie algorithm. *Journal of Chemical Physics*, 118(11):4999–5010, 2003.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1:pp. 162–170, 2000.

[5] C. Baier, B. Haverkort, H. Hermann, and J. P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. on Software Eng.*, Vol. 29(6):pp. 524–541, June 2003.

[6] P. Ballarini, T. Mazza, A. Palmisano, and A. Csikász-Nagy. Studying irreversible transitions in a model of cell cycle regulation. *Electronic Notes in Theoretical Computer Science*, (232):39–53, 2009.

[7] M. Barberis, E. Klipp, M. Vanoni, and L. Alberghina. Cell size at S phase initiation: an emergent property of the G1/S network. *PLoS Computational Biology*, 3, 2007.

[8] D. Barik, W.T. Baumann, M.R. Paul, B. Novak, and J. J. Tyson. A model of yeast cell-cycle regulation based on multisite phosphorylation. *Molecular Systems Biology*, 6(1), 2010.

[9] D. Battogtokh and J. J. Tyson. Bifurcation analysis of a model of the budding yeast cell cycle. *Chaos*, 14:653–661, 2004.

[10] J. M. Bean, E. D. Siggia, and F. R. Cross. Coherence and timing of cell cycle start examined at single-cell resolution. *Molecular Cell*, 21:3–14, 2006.

[11] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian Computation in Population Genetics. *Genetics*, 162(4):2025–2035, 2002.

[12] L. Bortolussi and A. Policriti. Stochastic concurrent constraint programming and differential equations. *Electronic Notes in Theoretical Computer Science*, 190(3):27 – 42, 2007. Proceedings of QAPL 2007.

[13] J. T. Bradley, S. T. Gilmore, and J. Hillston. Analysing distributed internet worm attacks using continuous state-space approximation of process algebra models. *J. Comput. Syst. Sci.*, 74(6):1013–1032, 2008.

[14] S. Braunewell and S. Bornholdt. Superstability of the yeast cell cycle dynamics: ensuring causality in the presence of biochemical stochasticity. *Journal of Theoretical Biology*, 245(638), 2007.

[15] O. Bretscher. *Linear algebra with applications*. Prentice Hall, 2001.

[16] F. J. Bruggeman and H. V. Westerhoff. The nature of systems biology. *Trends in Microbiology*, 15:45–50, 2006.

[17] R. Bundschuh, F. Hayot, and C. Jayaprakash. Fluctuations and slow variables in genetic networks. *Biophysical Journal*, 84(3):1606 – 1615, 2003.

[18] M. Calder and J. Hillston. Process algebra modelling styles for biomolecular processes. *Lecture Notes in Computer Science*, 5750:1–25, 2009.

[19] L. Calzone, K. C. Chen, J. Zwolak, and J. J. Tyson. Budding yeast cell cycle homepage. `http://mpf.biol.vt.edu/research/budding_yeast_model/pp/`, 2005.

[20] L. Cardelli. Brane Calculi. In *Proc. Computational Methods in Systems Biology*, volume 3082 of *Lecture Notes in Computer Science*, pages 257–280. Springer, 2005.

[21] L. Cardelli. A process algebra master equation. In *QEST 2007*. IEEE Computer Society, 2007.

[22] L. Cardelli. From Processes to ODEs by Chemistry. In *IFIP TCS*, pages 261–281, 2008.

[23] L. Cardelli. On process rate semantics. *Theor. Comput. Sci.*, 391(3):190–215, 2008.

[24] L. Cardelli and A. D. Gordon. Mobile ambients. *Theor. Comput. Sci.*, 240(1):177–213, 2000.

[25] L. Cardelli and S. Pradalier. Where Membranes Meet Complexes. In *BioConcur 2005*, 2005.

[26] G. Casella and R. L. Berger. *Statistical inference*. Duxbury, 2002.

[27] M. Cassman. Barriers to progress in systems biology. *Nature*, 438, 2005.

[28] G. Charvin, F. R. Cross, and E. D. Siggia. A microfluidic device for temporally controlled gene expression and long-term fluorescent imaging in unperturbed dividing yeast cells. *PloS one*, 3(1):1468, 2008.

[29] G. Charvin, F. R. Cross, and E. D. Siggia. Forced periodic expression of G1 cyclins phase-locks the budding yeast cell cycle. *Proceedings of the National Academy of Sciences*, 106(16):6632, 2009.

[30] K. C. Chen, L. Calzone, A. Csikász-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Molecular Biology of the Cell*, 15:3841–3862, 2004.

[31] K. C. Chen, A. Csikász-Nagy, B. Gyorffy, J. Val, B. Novak, and J. J. Tyson. Kinetic analysis of a molecular model of the budding yeast cell cycle. *Molecular Biology of the Cell*, 11:369–391, 2000.

[32] D. Chiarugi, M. Curti, P. Degano, and R. Marangoni. VICE: A VIrtual CEll. In *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2005.

[33] I. C. Chou, H. Martens, and E. O. Voit. Parameter estimation in biochemical systems models with alternating regression. *Theoretical Biology and Medical Modelling*, 3(25), 2006.

[34] K. Chu, Y. Deng, and J. Reinitz. Parallel simulated annealing by mixing of states. *Journal of Computational Physics*, 148:646–662, 1999.

[35] A. Ciliberto, F. Capuani, and J. J. Tyson. Modeling networks of coupled enzymatic reactions using the total quasi steady state approximation. *PLoS Computational Biology*, 3, 2007.

[36] F. Ciocchetta and J. Hillston. Bio-PEPA: an extension of the process algebra PEPA for biochemical networks. In *Proceedings of FBTC 2007*. ENTCS, 2007.

[37] F. Ciocchetta and J. Hillston. Bio-PEPA: a framework for the modelling and analysis of biochemical networks. *Theoretical Computer Science*, 410:3065–3084, 2009.

[38] E. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. MIT Press, 1999.

[39] A. Coletta, R. Gori, and F. Levi. Approximating Probabilistic Behaviors of Biological Systems Using Abstract Interpretation. *Electr. Notes Theor. Comput. Sci.*, 229(1):165–182, 2009.

[40] N. A. Cookson, S. W. Cookson, L. S. Tsimring, and J. Hasty. Cell cycle-dependent variations in protein concentration. *Nucleic Acids Research*, 2009.

[41] Y. Craciun, G. Tang and Feinberg M. Understanding bistability in complex enzyme-driven reaction networks. *Proc Natl Acad Sci USA*, 103(23), 2006.

[42] F. R. Cross, V. Archambault, M. Miller, and M. Klovstad. Testing a mathematical model for the yeast cell cycle. *Molecular Biology of the Cell*, 13:52–70, 2002.

[43] A. Csikász-Nagy. Computational systems biology of the cell cycle. *Briefings in Bioinformatics*, 10:424–434, 2009.

[44] A. Csikász-Nagy, D. Battogtokh, K. C. Chen, B. Novak, and J. J. Tyson. Analysis of a generic model of eukaryotic cell-cycle regulation. *Biophysical Journal*, 90:4361–4379, 2006.

[45] A. Csikász-Nagy, B. Novak, and J. J. Tyson. Reverse engineering models of cell cycle regulation. *Advances in Experimental Medicine and Biology*, 641:88–97, 2008.

[46] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-Based Modelling of Cellular Signalling. In *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 17–41. Springer, 2007.

[47] V. Danos and S. Pradalier. Projective Brane Calculus. In *CMSB*, volume 3082 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2004.

[48] G.S. Datta and M. Ghosh. On the invariance of noninformative priors. *The annals of Statistics*, 24(1):141–159, 1996.

[49] B. David and G. Bastin. A maximum likelihood parameter estimation method for nonlinear dynamicsl systems. In *38th Conference on Decision & Control*, Phoenix - Arizona USA, December 1999.

[50] M. P. C. David, J. Y. Bantang, and E. R. Mendoza. A Projective Brane Calculus with Activate, Bud and Mate as Primitive Actions. *T. Comp. Sys. Biology*, 11:164–186, 2009.

[51] U. de Lichtenberg, L. J. Jensen, A. Fausboll, T. S. Jensen, P. Bork, and S. Brunak. Comparison of computational methods for the identification of cell cycle regulated genes. 21(7):1164–1171, 2005.

[52] P. Degano, D. Prandi, C. Priami, and P. Quaglia. Beta-binders for Biological Quantitative Experiments. *Electr. Notes Theor. Comput. Sci.*, 164(3):101–117, 2006.

[53] L. Dematté, R. Larcher, A. Palmisano, C. Priami, and A. Romanel. *Programming Biology in BlenX.*

[54] L. Dematté and T. Mazza. On Parallel Stochastic Simulation of Diffusive Systems. *Proceedings of the sixth International Conference on Computational Methods in Systems Biology (CMSB2008)*, LNBI 5307:191 – 210, 2008.

[55] L. Dematté, C. Priami, and A. Romanel. The Beta Workbench: a computational tool to study the dynamics of biological systems. *Briefings in Bioinformatics*, 9(5):437–449, 2008.

[56] L. Dematté, C. Priami, and A. Romanel. *The BlenX Language: A Tutorial*, volume 5016, pages 313–365. LNCS, 2008.

[57] S. Di Talia, J. M. Skotheim, J. M. Bean, E. D. Siggia, and F. R. Cross. The effects of molecular noise and size control on variability in the budding yeast cell cycle. *Nature*, 448:947–951, 2007.

[58] J. Ding. *Structural and Fluid Analysis for Large Scale PEPA Models – With Applications to Content Adaptation Systems.* PhD thesis, School of Informatics, The University of Edinburgh, 2009.

[59] K. Entacher, A. Uhl, and S. Wegenkittl. Linear congruential generators for parallel Monte-Carlo: the Leap-Frog case. *Monte Carlo Methods and Applications*, 4(1):1–16, 1998.

[60] P. Érdi and J. Tóth. *Mathematical Models of Chemical Reactions.* Princeton University Press, 1989.

[61] B. Ermentrout. *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students.* Society for Industrial Mathematics, 2002.

[62] J. Evans and A. Rzhetsky. Machine science. *Science*, 329(5990):399–400, 2010.

[63] T. Evans, E. T. Rosenthal, J. Youngblom, D. Distel, and T. Hunt. Cyclin: a protein specified by maternal mRNA in sea urchin eggs that is destroyed at each cleavage division. *Cell*, 33:389–396, 1983.

[64] G. C. Ewing, D. McNickle, and L. Pawlikowski. Multiple replications in parallel: Distributed generation of data for speeding up quantitative stochastic simulation. *Proceedings of the 15th Congress of Int. Association for Matemathics and Computer in Simulation*, 1997.

[65] A. Faure and D. Thieffry. Logical modelling of cell cycle control in eukaryotes: a comparative study. *Mol Biosyst*, 5:1569–1581, 2009.

[66] M. Feinberg. *Chem. Eng. Sci*, (42), 1987.

[67] J. Fisher and T. A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, 2007.

[68] W. Fontana and L. W. Buss. *The Barrier of Objects: From Dynamical Systems to Bounded Organizations*, pages 56–116. Addison-Wesley, 1996.

[69] M. Forlin, T. Mazza, and D. Prandi. Predicting the effects of parameters changes in stochastic models through parallel synthetic experiments and multivariate analysis. *Proceedings of HiBi 2010, IEEE Computer Society CPS*, 2010.

[70] N. Friedman, M. Linial, I. Nachman, and D. Peer. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3):601–620, 2000.

[71] H. Fuss, W. Dubitzky, C. S. Downes, and M. J. Kurth. Mathematical models of cell cycle regulation. *Briefings in Bioinformatics*, 6:163–177, 2005.

[72] N. P. Gauthier, M. E. Larsen, R. Wernersson, U. De Lichtenberg, L. J. Jensen, S. Brunak, and T. S. Jensen. Cyclebase.org - a comprehensive multi-organism online database of cell-cycle experiments. *Nucleic acids research*, 36(suppl 1):D854, 2008.

[73] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104(9):1876–1889, 2000.

[74] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[75] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115:1716, 2001.

[76] P. W. Glynn and P. Heidelberger. Analysis of parallel replicated simulations under a completion time constraint. *ACM TOMACS*, 1(1):3–23, 1991.

[77] G. Goel, I-C. Chou, and E. O. Voit. System estimation from metabolic time-series data. *Bioinformatics*, 24(21):2505–2511, 2008.

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

[79] A. Goldbeter. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proceedings of the National Academy of Sciences USA*, 88:9107–9111, 1991.

[80] I. Golding, J. Paulsson, S. M. Zawilski, and E. C. Cox. Real-time kinetics of gene activity in individual bacteria. *Cell*, pages 1025–1036, 123 2005.

[81] A. Golightly and D. J. Wilkinson. Bayesian sequential inference for stochastic kinetic biochemical network models. *Journal of Computational Biology*, 13(3):838–851, 2006.

[82] D. Gonze, J. Halloy, and A. Goldbeter. Deterministic Versus Stochastic Models for Circadian Rhythms. *J. of Biol. Phys.*, 28, 2002.

[83] J. Goutsias. A hidden Markov model for transcriptional regulation in single cells. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, 3(1):57–71, 2006.

[84] H.L. Harney. *Bayesian inference: parameter estimation and decisions*. Springer Verlag, 2003.

[85] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature*, 402:C47C52, 1999.

[86] L. H. Hartwell, R. K. Mortimer, J. Culotti, and M. Culotti. Genetic control of the cell division cycle in yeast: V. genetic analysis of cdc mutants. *Genetics*, 74:267–286, 1973.

[87] L. H. Hartwell and T. A. Weinert. Checkpoints: controls that ensure the order of cell cycle events. *Science*, 246:629–634, 1989.

[88] R. A. Hayden and J. T. Bradley. A fluid analysis framework for a Markovian process algebra. *Theor. Comput. Sci.*, 411(22-24):2260–2297, 2010.

[89] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 391:239–257, 2008.

[90] M. Heiner, D. Gilbert, and R. Donaldson. Petri Nets for Systems and Synthetic Biology. In *SFM*, pages 215–264, 2008.

[91] J. Hillston. *A Compositional Approach to Performance Modelling*. PhD thesis, University of Edinburgh, 1994.

[92] J. Hillston. Fluid Flow Approximation of PEPA models. In *QEST '05: Proceedings of the Second International Conference on the Quantitative Evaluation of Systems*, page 33. IEEE Computer Society, 2005.

[93] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Sci STKE*, 2006(344), 2006.

[94] W. S. Hlavacek and M. A. Savageau. Rules for coupled expression of regulator and effector genes in inducible circuits. *J. Mol. Biol.*, 255:121–139, 1996.

[95] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, and N. Simus. Copasi - a complex pathway simulator. bioinformatics. *Bioinformatics*, 22:3067–3074, 2006.

[96] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI - a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, 2006.

[97] A. E. C. Ihekwaba, D. S. Broomhead, R. Grimley, N. Benson, and D. B. Kell. Iκbα. *Systems Biology*, 1:99–103, 2004.

[98] A. E. C. Ihekwaba, D. S. Broomhead, R. Grimley, N N. Benson, M. R. H. White, and D. B. Kell. Iκbα. *IEE Proceedings Systems Biology*, 152:153–160, 2005.

[99] A. E. C. Ihekwaba, S. J. Wilkinson, D. S. Broomhead, D. Waithe, R. Grimley, N. Benson, and D. B. Kell. Bridging the gap between in silico and cell based analysis of the NF-κB signalling pathway by in vitro studies of IKK2. *FEBS Journal*, 27:1678–1690, 2007.

[100] E.T. Jaynes and G.L. Bretthorst. *Probability theory: the logic of science*. Cambridge Univ Pr, 2003.

[101] S. Kar, W. T. Baumann, M. R. Paul, and J. J. Tyson. Exploring the roles of noise in the eukaryotic cell cycle. *Proc Natl Acad Sci USA*, 106:6471–6476, 2009.

[102] M. B. Kastan and J. Bartek. Cell-cycle checkpoints and cancer. *Nature*, 432:316–323, 2004.

[103] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.

[104] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita. Dynamic modeling of genetic networks using genetic algorithm and s-system. *Bioinformatics*, 10(5):643–650, 2003.

[105] H. Kitano. Systems biology: a brief overview. *Science*, 295(5560):1662–1664, March 2002.

[106] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. *Systems biology in practice: concepts, implementation and application*. Wiley-VCH, 2005.

[107] A. L. Koch and M. Schaechter. A model for statistics of the cell division process. *J Gen Microbiol*, 29:435–454, 1962.

[108] C. Kuttler and J. Niehren. Gene regulation in the pi-calculus: simulating cooperativity at the lambda switch. *Transactions on Computational Systems Biology VII*, 4230:24–55, 2006.

[109] M. Kwiatkowska and I. Stark. The continuous pi-calculus: A process algebra for biochemical modelling. In *CMSB*, volume 5307 of *Lecture Notes in Computer Science*, pages 103–122. Springer, 2008.

[110] M. Z. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with prism: A hybrid approach. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 52–66. Springer-Verlag, 2002.

[111] M. Kwiatkowski. *A formal computational framework for the study of molecular evolution*. PhD thesis, School of Informatics, The University of Edinburgh, 2010.

[112] C. Laneve and F. Tarissan. A simple calculus for proteins and cells. *Theor. Comput. Sci.*, 404(1-2):127–141, 2008.

[113] P. Langevin. On the theory of brownian motion. *C. R. Acad. Sci.*, 146:530–533, 1908.

[114] Y. Lazebnik. Can a biologist fix a radio?–or, what I learned while studying apoptosis. *Cancer Cell*, 2(3):179 – 182, 2002.

[115] N. Le Novère and T. S. Shimizu. STOCHSIM: modelling of stochastic biomolecular processes. *Bioinformatics*, 17(6):575–576, 2001.

[116] P. Lecca and Priami C. Cell cycle control in eukaryotes: a biospi model. 3:51–63, 2003.

[117] P. Lecca, A. Palmisano, and A. E. Ihekwaba. Correlation-based network inference and modelling in systems biology: the NF-κB signalling network case study. In *Int. Conf. on Intelligent Systems, Modelling and Simulation (ISMS2010)*, pages 170–175. IEEE Computer Society, 2010.

[118] P. Lecca, A. Palmisano, and A. E. Ihekwaba. Inferring the kinetic constants of cyclin-triggered expression of cdc20 gene in eukaryotic cell cycle. *Online Journal of Bioinformatics*, 11(2):177–216, 2010.

[119] P. Lecca, A. Palmisano, A. E. Ihekwaba, and C. Priami. Calibration of dynamic models of biological systems with kinfer. *European Biophysics Journal*, 2009.

[120] P. Lecca, A. Palmisano, and C. Priami. Deducing chemical reaction rate constants and their regions of confidence from noisy measurements of time series of concentration. In *11th Int. Conference on Computer Modelling and Simulation (UKSim 2009)*, 2009.

[121] P. Lecca, C. Priami, P. Quaglia, B. Rossi, C. Laudanna, and G. Constantin. A Stochastic Process Algebra Approach to Simulation of Autoreactive Lymphocyte Recruitment. *Simulation*, 80(6):273–288, 2004.

[122] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang. The yeast cell-cycle network is robustly designed. *Proc Natl Acad Sci USA*, 101:47814786, 2004.

[123] M. P. Little, W. F. Heidenreich, and G. Li. Parameter identifiability and redundancy: Theoretical considerations. *PloS ONE*, 5(1), 2010.

[124] L. M. Loew and J. C. Schaff. The Virtual Cell: a software environment for computational cell biology. *Trends in Biotechnology*, 19(10):401–406, 2001.

[125] P. G. Lord and A. E. Wheals. Asymmetrical division of saccharomyces cerevisiae. *The Journal of Bacteriology*, 142:808818, 1980.

[126] A. Marin-Sanguino, E. O. Voit, C. Gonzalez-Alcon, and N. V. Torres. Optimization of biotechnological systems through geometric programming. *Theoretical Biology and Medical Modelling*, 4(38), 2007.

[127] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavare. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences of the United States of America*, 100(26):15324–15328, 2003.

[128] The MathWorks and NIST. JAMA: A java matrix package. `http://math.nist.gov/javanumerics/jama`, 2005.

[129] P. Mendes. GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Comput Appl Biosci*, 9(5):563–571, 1993.

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

[131] R. Milner. *Communicating and Mobile Systems: the Pi-Calculus*. Cambridge University Press, 1999.

[132] C. G. Moles, P. Mendes, and J. R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res*, 13(11):2467–74, 2003.

[133] G. C. Moles, P. Mendes, and J. R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Res.*, 13:2467–2474, 2003.

[134] P. T. Monteiro, D. Ropers, R. Mateescu, A. T. Freitas, and H. de Jong. Temporal logic patterns for querying dynamic models of cellular interaction networks. *Bioinformatics*, 24:227–233, 2008.

[135] I. Mura and A. Csikász-Nagy. Stochastic Petri Net extension of a yeast cell cycle model. *Journal of Theoretical Biology*, 254:850–860, 2008.

[136] I. Mura, D. Prandi, C. Priami, and A. Romanel. Exploiting non-Markovian Bio-Processes. *Electronic Notes in Theoretical Computer Science*, 253(3):83–98, 2009.

[137] K. Nasmyth. At the heart of the budding yeast cell cycle. *Trends in Genetics*, 12(10):405–412, 1996.

[138] D. E. Nelson, A. E. C. Ihekwaba, M. Elliott, J. R. Johnson, C. A. Gibney, B. E. Foreman, G. Nelson, V. See, C. A. Horton, D. G. Spiller, S. W. Edwards, H. P. McDowell, J. F. Unitt, E. Sullivan, R. Grimley, N. Benson, D. Broomhead, D. B. Kell, and M. R. H. White. Oscillations in NF-κB signaling control the dynamics of gene expression. *Science*, 306:704–708, 2004.

[139] A. Neumann. Graphical Gaussian Shape Models and Their Application to Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):316–329, 2003.

[140] F. Nielson, H. R. Nielson, P. Priami, and D. Rosa. Control flow analysis for bioambients. *Electr. Notes Theor. Comput. Sci.*, 180(3):65–79, 2007.

[141] B. Novak, A. Csikász-Nagy, B. Gyorffy, K. Nasmyth, and J. J. Tyson. Model scenarios for evolution of the eukaryotic cell cycle. *Phil Trans R Soc Lond B*, Vol. 353:2063–2076, 1998.

[142] B. Novak and J. J. Tyson. *Cell Cycle Controls*, pages 261–284. Springer, 2003.

[143] P. Nurse. Genetic control of cell size at cell division in yeast. *Nature*, 256:547–551, 1975.

[144] P. Nurse. A long twentieth century of the cell cycle and beyond. *Cell*, 100(71), 2000.

[145] P. Nurse. Life, logic and information. *Nature*, 454, 2008.

[146] T. D. Panning, L. T. Watson, N. A. Allen, K. C. Chen, C. A. Shaffer, and J. J. Tyson. Deterministic parallel global parameter estimation for a model of the budding yeast cell cycle. *J. of Global Optimization*, 40(4):719–738, 2008.

[147] M. Pedersen. *Modular languages for systems and synthetic biology*. PhD thesis, School of Informatics, The University of Edinburgh, 2010.

[148] M. Pedersen and G. Plotkin. A language for biochemical systems. In *Computational Methods in Systems Biology*, pages 63–82. Springer, 2008.

[149] J. Pfanzagl and R. Hamboker. *Parametric Statistical Theory*. Walter de Gruyter, 1994.

[150] A. Phillips and L. Cardelli. Efficient, Correct Simulation of Biological Processes in the Stochastic Pi-calculus. In *CMSB*, volume 4695 of *LNCS*, page 184. Springer, 2007.

[151] P. K. Polisetty and E. O. Voit. Identification of metabolic system parameters using global optimization methods. *Theoreteical Biology and Medical Modelling*, 3(4), 2006.

[152] D. Prandi, C. Priami, and P. Quaglia. Communicating by compatibility. *JLAP*, 75:167, 2008.

[153] C. Priami. Stochastic π-Calculus. *The Computer Journal*, 38:578–589, 1995.

[154] C. Priami. Algorithmic systems biology. an opportunity for computer science. *Communications of the ACM*, 2009.

[155] C. Priami and P. Quaglia. Beta Binders for Biological Interactions. In *CMSB*, pages 20–33, 2004.

[156] C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, 2001.

[157] J. Puchalka and A. M. Kierzek. Bridging the gap between stochastic and deterministic regimes in the kinetic simulations of the biochemical reaction networks. *Biophysical Journal*, 86(3):1357–1372, 2004.

[158] M. Quach, N. Brunel, and F. d'Alché Buc. Estimating parameters and hidden variables in non-linear state-space models based on ODEs for biological networks inference. *Bioinfromatics*, 23:3209–16, 2007.

[159] E. Queralt, C. Lehane, B. Novak, and F. Uhlmann. Downregulation of pp2a(cdc55) phosphatase by separase initiates mitotic exit in budding yeast. *Cell*, 125(719), 2006.

[160] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theor. comput. Sci.*, 325(1):141–167, 2004.

[161] A. Regev and E. Shapiro. Cells as Computations. *Nature*, 419:343, 2002.

[162] J. Reinitz, E. Mjolsness, and D. H. Sharp. Model for cooperative control of positional information in drosophila by bicoid and maternal hunchback. *Journal of Experimental Zoology*, 271:47–56, 1995.

[163] S. Reinker, R. M. Altman, and J. Timmer. Parameter estimation in stochastic biochemical reactions. In *IEEE Proc. Syst. Biol*, volume 153, 2006.

[164] M. Rodrigez-Fernandez, P. Mendes, and J. Banga. A hybrid approach for efficient and robust parameter estimation in biochemical pathways. *BioSystems*, 83:248–265, 2006.

[165] R. Ronquist. Bayesian inference. Technical report, BCS 5936-Fall, 2005.

[166] M. Saccomani, S. Audoly, G. Bellu, and L. D'Angio. Parameter identifiability of nonlinear biological systems. *Lecture Notes in Control and Information Sciences*, 294, 2004.

[167] M. Savageau. *Coupled circuits of gene regulation.* Sequence specificity in transcription and translation. 1985.

[168] M. A. Savageau and P. J. Sands. *Completely uncoupled and perfectly coupled circuits for inducible gene regulation.* Canonical non-linear modeling: S-system approach to understanding complexity. E. O. Voit eds, Van Nostrand Reinhold, New York, 1985.

[169] M. A. Savageau and E. O. Voit. Power-law approach to modeling biological-systems theory. *J. Ferment. Technol.*, 60:221–228, 1982.

[170] M. Schleiden. Beitrage fur phytogenesis. *J. Arch. Anat. Physiol. Wiss. Med.*, 13:137–176, 1838.

[171] T. Schwann. *Mikroskopische Untersuchungen uber die ubereinstimmung in der Struktur und dem Wachstum der Tiere und Pflanzen.* Sander'schen Buchhandlung, Berlin, 1839.

[172] C. A. Shaffer, J. W. Zwolak, R. Randhawa, and J. J. Tyson. Modeling molecular regulatory networks with JigCell and PET. *Methods in molecular biology (Clifton, NJ)*, 500:81, 2009.

[173] J. C. Sible and J. J. Tyson. Mathematical modeling as a tool for investigating cell cycle control networks. *Methods*, 41:238–247, 2007.

[174] S. A. Sisson, Y. Fan, and Mark M. Tanaka. Sequential Monte Carlo without likelihoods. 104(6):1760–1765, 2007.

[175] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[176] K. Sriram, G. Bernot, and F. Kepes. A minimal mathematical model combining several regulatory cycles from the budding yeast cell cycle. *IET System Biology*, 1:326–341, 2007.

[177] J. Stelling and E. D. Gilles. Mathematical modeling of complex regulatory networks. *IEEE Trans Nanobioscience*, 3:172–179, 2004.

[178] W. J. Stewart. *Introduction to numerical solution of Markov Chains.* Princeton, 1994.

[179] M. Sugimoto, S. Kikuchi, and M. Tomita. Reverse engineering of biochemical equations from time-course data by means of genetic programming. *BioSystems*, 80:155–164, 2005.

[180] D. Thieffry. Dynamical roles of biological regulatory circuits. *Briefings in Bioinformatics*, 8:220–225, 2007.

[181] C. D. Thron. Mathematical analysis of a model of the mitotic clock. *Science*, 254:122–123, 1991.

[182] T. Tian, S. Xu, and K. Burrage. Simulated maximum likelihood method for estimating kinetic rates in gene expression. *Bioinformatics*, 23(1):84–91, 2007.

[183] R. Tibshirani. Noninformative priors for one parameter of many. *Biometrika*, 76(3):604, 1989.

[184] M. Tomita, K. Hashimoto, K. Takahashi, T. S. Shimizu, Y. Matsuzaki, F. Miyoshi, K. Saito, S. Tanida, K. Yugi, J. C. Venter, and C. Hutchison. E-CELL: software environment for whole-cell simulation. *Bioinformatics*, 15(1):72–84, January 1999.

[185] M. Tribastone, S. Gilmore, and J. Hillston. Scalable Differential Analysis of Process Algebra Models, 2010. To appear in *Transactions on Software Engineering*.

[186] J. J. Tyson. Modeling the cell division cycle: cdc2 and cyclin interactions. *Proceedings of the National Academy of Sciences USA*, 88:7328–7332, 1991.

[187] J. J. Tyson, K. Chen, and B. Novak. Network dynamics and cell physiology. *Nat Rev Mol Cell Biol*, 2:908–916, 2001.

[188] J. J. Tyson, A. Csikász-Nagy, and B. Novak. The dynamics of cell cycle regulation. *Bioessays*, 24(12):1095–1109, 2002.

[189] J. J. Tyson and B. Novak. Temporal organization of the cell cycle. *Current Biology*, 18:R759–R768, 2008.

[190] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[191] E. O. Voit and J. Almeida. Decoupling dynamical systems for pathway identification from metabolic profiles. *Bioinformatics*, 20:1670–1681, 2004.

[192] E. O. Voit, J. Almeida, S. Marino, R. Lall, G. Goel, A. R. Neves, and H. Santos. Regulation of glycolysis in lactococcus lactis: an unfinisched systems biological case study. *IEE Proc.-Syst. Biol.*, 153(4):286–298, 2006.

[193] V. Vyshemirsky and M. A. Girolami. Bayesian ranking of biochemical system models. *Bioinformatics*, 24(6):833–839, 2008.

[194] Y. Wang, S. Christley, E. Mjolsness, and X. Xie. Parameter inference for discretely observed stochastic kinetic models using stochastic gradient descent. *BMC Systems Biology*, 4(99), 2010.

[195] D. J. Wilkinson. Bayesian methods in bioinformatics and computational systems biology. *Briefings in bioinformatics*, 8(2):109, 2007.

[196] R. D. Wilkinson. Approximate bayesian computation (abc) gives exact results under the assumption of model error. *Biometrika*, pages 1–13, 2008.

[197] J. C. Wooley and Lin H. *Computational modeling and simulation as enablers for biological discovery (in Catalyzing Inquiry at the Interface of Computing and Biology)*. National Academies Press, 2005.

[198] A. M. Zhabotinsky. A history of chemical oscillations and waves. *Chaos*, 1:279–386, 1991.

[199] Z. Zi and E. Klipp. SBML-PET: a Systems Biology Markup Language-based parameter estimation tool. *Bioinformatics*, 22(21):2704, 2006.

[200] J.W. Zwolak, J.J. Tyson, and L.T. Watson. Parameter estimation for a mathematical model of the cell cycle in frog eggs. *Journal of computational biology*, 12(1):48–63, 2005.

Appendix A

A.1 BlenX model of Budding Yeast Cell Cycle

Here we report the BlenX code used in Sections 4.4-4.5 for perfoming stochastic simulations of the Chen model.

The .prog and .types file contain the definition of the structure of the modelled chemical reaction network, while the .func file contains the definition of the kinetic constants and rate functions. Just the parameter set of the wild type model in glucose is reported below, because it is possible to obtain the input file of all the other mutants, just changing the appropriate constants in the .func file, according to the assignments on the web-site `http://mpf.biol.vt.edu/research/budding_yeast_model/pp/`.

Note that the value of the "alpha" constant (used for converting the concentration levels to molecular numbers) has been calculated in such a way that the total amount of proteins in the simulations is comparable to the experimental data presented in [101].

A.1.1 .prog file

```
[steps = 1000, delta = 0.5]
<<BASERATE:inf>>

template spont_degradation : pproc<<rate kin_rate>> = die(rate(kin_rate));
template receiving_degr_signal : pproc<<name channel>> = channel?().die;
template guarded_spont_degradation : pproc<<name N1, binder state, rate kin_rate>> =
                                    (if((N1,state)) then die(rate(kin_rate)) endif);

let C2 : bproc = #(c2_state, C2_ty)     [ receiving_degr_signal<<c2_state>>  ];
let C2P : bproc = #(c2p_state, C2P_ty) [ receiving_degr_signal<<c2p_state>> ];
let C5 : bproc = #(c5_state, C5_ty)     [ receiving_degr_signal<<c5_state>>  ];
let C5P : bproc = #(c5p_state, C5P_ty)  [ receiving_degr_signal<<c5p_state>> ];

let CDC14 : bproc = #(cdc14_state, CDC14_ty), #(cdc14_out, CDC14_OUT_ty)
[ receiving_degr_signal<<cdc14_state>> |  spont_degradation<<kd14>> | rep cdc14_out!().nil ];

let activation_CDC15 : pproc =
        if(cdc15_degr, CDC15I_ty) then cdc15_act?().ch(cdc15_degr, CDC15_ty).rec?().nil endif;
let inactivation_CDC15 : pproc =
```

```
        if(cdc15_degr, CDC15_ty) then ch(rate(ki15),cdc15_degr,CDC15I_ty).rec2?().nil endif;

let Cdc15_process : pproc =
receiving_degr_signal<<cdc15_degr>> |
activation_CDC15 | rep rec!().activation_CDC15 |
  inactivation_CDC15 | rep rec2!().inactivation_CDC15;

let CDC15 : bproc = #(cdc15_degr, CDC15_ty),   #(cdc15_act, CDC15_ACT_ty) [ Cdc15_process ];
let CDC15i : bproc = #(cdc15_degr, CDC15I_ty), #(cdc15_act, CDC15_ACT_ty) [ Cdc15_process ];

let send_degradation_signal : pproc =
                              if(cdc20_degr, CDC20_ty) then cdc20_out!().rec2!().nil endif;

let self_activation : pproc = if(cdc20_degr, CDC20I_ty) then
                              ch(rate(ka20_p),cdc20_degr,CDC20_ty).hide(cdc20_act).rec!().nil endif;
let activation_by_iep : pproc = if(cdc20_degr,CDC20I_ty) then
                             cdc20_act?().ch(cdc20_degr,CDC20_ty).hide(cdc20_act).rec3!().nil endif;

let Cdc20_process : pproc =
   receiving_degr_signal<<cdc20_degr>> | spont_degradation<<kd20>> |
   self_activation | rep rec?().self_activation | send_degradation_signal | rep rec2?().send_degradation_signal |
   activation_by_iep | rep rec3?().activation_by_iep;

let CDC20 : bproc = #(cdc20_degr, CDC20_ty),
                    #(cdc20_out, CDC20_out_ty), #h(cdc20_act, CDC20_ACT_ty) [ Cdc20_process ];
let CDC20i : bproc = #(cdc20_degr, CDC20I_ty),
                     #(cdc20_out, CDC20_out_ty), #(cdc20_act, CDC20_ACT_ty) [ Cdc20_process ];

let CDH1 : bproc = #(cdh1_ch, CDH1_ty),#(cdh1_4killer, CDH1_4KILLER_ty)
[ receiving_degr_signal<<cdh1_4killer>> |
   rep cdh1_ch!().nil | spont_degradation<<kdcdh>> ];
let CDH1i : bproc = #(cdh1i_ch, CDH1I_ty),#(cdh1_4killer, CDH1_4KILLER_ty)
[ receiving_degr_signal<<cdh1_4killer>> |
  spont_degradation<<kdcdh>>  ];

let CLB2 : bproc = #(clb2_ch, CLB2_ty)
  [ receiving_degr_signal<<clb2_ch>> | spont_degradation<<kdb2_p>> | rep clb2_ch!().nil ];

let CLB5 : bproc = #(clb5_ch, CLB5_ty)
[ receiving_degr_signal<<clb5_ch>> | spont_degradation<<kdb5_p>> ];

let CLN2 : bproc = #(cln2_ch, CLN2_ty)
  [ receiving_degr_signal<<cln2_ch>> | spont_degradation<<kdn2>>  ];

let ESP1 : bproc = #(esp1_ch, ESP1_ty) [ receiving_degr_signal<<esp1_ch>> ];

let F2  : bproc = #(f2_ch, F2_ty)   [ receiving_degr_signal<<f2_ch>> ];
let F2P : bproc = #(f2p_ch, F2P_ty) [ receiving_degr_signal<<f2p_ch>> ];

let F5  : bproc = #(f5_ch, F5_ty)   [ receiving_degr_signal<<f5_ch>> ];
let F5P : bproc = #(f5p_ch, F5P_ty) [ receiving_degr_signal<<f5p_ch>> ];

let IE  : bproc = #(ie_ch, IE_ty)   [ receiving_degr_signal<<ie_ch>> ];
let IEP : bproc = #(iep_ch, IEP_ty) [ receiving_degr_signal<<iep_ch>> | rep iep_ch!().nil ];
```

```
let NET1 : bproc = #(net1_degr, NET1_ty), #(net1_deph_ch, NET1P_DEPH_ty)
[ receiving_degr_signal<<net1_degr>> | spont_degradation<<kdnet>> ];

let NET1P : bproc = #(net1_degr, NET1P_ty), #(net1_deph_ch, NET1P_DEPH_ty)
[ receiving_degr_signal<<net1_degr>> | spont_degradation<<kdnet>> |
 (ch(rate(kppnet_p),net1_degr,NET1_ty).nil + net1_deph_ch?().ch(net1_degr,NET1_ty).nil) ];

let PDS1 : bproc = #(pds1_ch, PDS1_ty)
  [ receiving_degr_signal<<pds1_ch>> | spont_degradation<<kd1pds_p>> ];

let PE : bproc = #(pe_ch, PE_ty) [ receiving_degr_signal<<pe_ch>> ];

let PPX : bproc = #(ppx_ch, PPX_ty) [ receiving_degr_signal<<ppx_ch>> | rep ppx_ch!().nil ];

let RENT : bproc = #(rent_degr, RENT_ty), #(rent_deph_ch, RENT_DEPH_ty)
[ receiving_degr_signal<<rent_degr>> ];

let RENTP : bproc = #(rent_degr, RENTP_ty), #(rent_deph_ch, RENT_DEPH_ty)
[ receiving_degr_signal<<rent_degr>> |
  (ch(rate(kppnet_p),rent_degr,RENT_ty).nil + rent_deph_ch?().ch(rent_degr,RENT_ty).nil) ];

let SIC1 : bproc = #(sic1_degr, SIC1_ty), #(sic1p_deph_ch, SIC1P_DEPH_ty)
[ receiving_degr_signal<<sic1_degr>> |
  guarded_spont_degradation<<sic1_degr,SIC1P_ty,kd3c1>> ];

let dephosphorilation_SIC1P : pproc = sic1p_deph_ch?().ch(sic1_degr,SIC1_ty).nil;

let SIC1P : bproc = #(sic1_degr, SIC1P_ty), #(sic1p_deph_ch, SIC1P_DEPH_ty)
[ receiving_degr_signal<<sic1_degr>> |
  guarded_spont_degradation<<sic1_degr,SIC1P_ty,kd3c1>> | dephosphorilation_SIC1P ];

let CDC6 : bproc = #(cdc6_degr, CDC6_ty), #(cdc6p_deph_ch, CDC6P_DEPH_ty)
[ receiving_degr_signal<<cdc6_degr>> |
  guarded_spont_degradation<<cdc6_degr,CDC6P_ty,kd3f6>> ];

let dephosphorilation_CDC6P : pproc = cdc6p_deph_ch?().ch(cdc6_degr,CDC6_ty).nil;
let CDC6P : bproc = #(cdc6_degr, CDC6P_ty), #(cdc6p_deph_ch, CDC6P_DEPH_ty)
[ receiving_degr_signal<<cdc6_degr>> |
  guarded_spont_degradation<<cdc6_degr,CDC6P_ty,kd3f6>> | dephosphorilation_CDC6P ];

let phosphorilation_dephosphorilation_SWI5 : pproc =
  swi5_deph_ch?().(
    if(swi5_degr, SWI5_ty)
      then  ch(swi5_degr,SWI5P_ty).ch(swi5_deph_ch,SWI5_DEPH_ty).rec!().nil   endif
+ if(swi5_degr, SWI5P_ty)
  then  ch(swi5_degr,SWI5_ty).ch(swi5_deph_ch,SWI5_PH_ty).rec!().nil   endif
);

let SWI5 : bproc = #(swi5_degr, SWI5_ty), #(swi5_deph_ch, SWI5_PH_ty)
[ receiving_degr_signal<<swi5_degr>> |
  spont_degradation<<kdswi>> |
  rep rec?().phosphorilation_dephosphorilation_SWI5 | phosphorilation_dephosphorilation_SWI5 ];

let SWI5P : bproc = #(swi5_degr, SWI5P_ty), #(swi5_deph_ch, SWI5_DEPH_ty)
[ receiving_degr_signal<<swi5_degr>> |
```

```
  spont_degradation<<kdswi>> |
  rep rec?().phosphorilation_dephosphorilation_SWI5 | phosphorilation_dephosphorilation_SWI5 ];

let TEM1GDP : bproc = #(tem1gdp_ch, TEM1GDP_ty)
[ receiving_degr_signal<<tem1gdp_ch>> | rep tem1gdp_ch!().nil ];

let TEM1GTP : bproc = #(tem1gtp_ch, TEM1GTP_ty)
  [ receiving_degr_signal<<tem1gtp_ch>> | rep tem1gtp_ch!().nil ];

when(CLN2 : : Synthesis_of_CLN2) new(1);
when(CLB2 : : Synthesis_of_CLB2) new(1);
when(CLB5 : : Synthesis_of_CLB5) new(1);
when(SIC1 : : Synthesis_of_SIC1) new(1);
when(SIC1 : : Phosphorylation_of_SIC1) split(Nil, SIC1P);
when(SIC1,CLB2 : : Assoc_of_CLB2_and_SIC1) join(C2);
when(C2 : : Dissoc_of_CLB2SIC1_complex) split(SIC1, CLB2);
when(SIC1,CLB5 : : Assoc_of_CLB5_and_SIC1) join(C5);
when(C5 : : Dissoc_of_CLB5SIC1) split(SIC1, CLB5);
when(C2 : : Phosphorylation_of_C2) split(Nil, C2P);
when(C2P : : Dephosphorylation_of_C2P) split(Nil, C2);
when(C5 : : Phosphorylation_of_C5) split(Nil, C5P);
when(C5P : : Dephosphorylation_of_C5P) split(Nil, C5);
when(C2 : : Degradation_of_CLB2_in_C2) split(Nil, SIC1);
when(C5 : : Degradation_of_CLB5_in_C5) split(Nil, SIC1);
when(C2P : : Degradation_of_SIC1_in_C2P) split(Nil, CLB2);
when(C5P : : Degradation_of_SIC1P_in_C5P_) split(Nil, CLB5);
when(C2P : : Degradation_of_CLB2_in_C2P) split(Nil, SIC1P);
when(C5P : : Degradation_of_CLB5_in_C5P) split(Nil, SIC1P);
when(CDC6 : : CDC6_synthesis) new(1);
when(CDC6 : : Phosphorylation_of_CDC6) split(Nil, CDC6P);
when(CDC6,CLB2 : : CLB2CDC6_complex_formation) join(F2);
when(F2 : : CLB2CDC6_dissociation) split(CDC6, CLB2);
when(CDC6,CLB5 : : CLB5CDC6_complex_formation) join(F5);
when(F5 : : CLB5CDC6_dissociation) split(CDC6, CLB5);
when(F2 : : F2_phosphorylation) split(Nil, F2P);
when(F2P : : F2P_dephosphorylation) split(Nil, F2);
when(F5 : : F5_phosphorylation) split(Nil, F5P);
when(F5P : : F5P_dephosphorylation) split(Nil, F5);
when(F2 : : CLB2_degradation_in_F2) split(Nil, CDC6);
when(F5 : : CLB5_degradation_in_F5) split(Nil, CDC6);
when(F2P : : CDC6_degradation_in_F2P) split(Nil, CLB2);
when(F5P : : CDC6_degradation_in_F5P) split(Nil, CLB5);
when(F2P : : CLB2_degradation_in_F2P) split(Nil, CDC6P);
when(F5P : : CLB5_degradation_in_F5P) split(Nil, CDC6P);
when(SWI5 : : Synthesis_of_SWI5) new(1);
when(IE : : Activation_of_IEP) split(Nil, IEP);
when(IEP : : Inactivation_1_IEP) split(Nil, IE);
when(CDC20i : : Synthesis_of_inactive_CDC20) new(1);
when(CDC20 : : Inactivation_2_CDC20) split(Nil, CDC20i);
when(CDH1 : : CDH1_synthesis) new(1);
when(CDH1i : : CDH1i_activation) split(Nil, CDH1);
when(CDH1 : : Inactivation_3_CDH1) split(Nil, CDH1i);
when(CDC14 : : CDC14_synthesis) new(1);
when(NET1,CDC14 : : Assoc_with_NET1_to_form_RENT) join(RENT);
when(RENT : : Dissoc_from_RENT) split(CDC14, NET1);
```

```
when(NET1P,CDC14 : : Assoc_with_NET1P_to_form_RENTP) join(RENTP);
when(RENTP : : Dissoc_from_RENP) split(NET1P, CDC14);
when(NET1 : : Net1_synthesis) new(1);
when(NET1 : : NET1_phosphorylation) split(Nil, NET1P);
when(RENT : : RENT_phosphorylation) split(Nil, RENTP);
when(RENT : : Degradation_of_NET1_in_RENT) split(Nil, CDC14);
when(RENTP : : Degradation_of_NET1P_in_RENTP) split(Nil, CDC14);
when(RENT : : Degradation_of_CDC14_in_RENT) split(Nil, NET1);
when(RENTP : : Degradation_of_CDC14_in_RENTP) split(Nil, NET1P);
when(TEM1GDP : : TEM1_activation) split(Nil, TEM1GTP);
when(TEM1GTP : : inactivation_1_TEM1GTP) split(Nil, TEM1GDP);
when(PPX : : PPX_synthesis) new(1);
when(PPX : : degradation_1_PPX) delete(1);
when(PDS1 : : PDS1_synthesis) new(1);
when(PE : : Degradation_of_PDS1_in_PE) split(Nil, ESP1);
when(ESP1,PDS1 : : Assoc_with_ESP1_to_form_PE) join(PE);
when(PE : : Disso_from_PE) split(ESP1, PDS1);

when(: CLB2_CLB5_KEZ2 <- 10, CLB2_CLB5_KEZ2 <- 9.95 :) update(ORI , reset_ORI_ORI_var);
when(: CLB2_KEZ -> 10, CLB2_KEZ <- 9.95 :) update(BUD , cell_division_BUD_var);
when(: CLB2_KEZ -> 10, CLB2_KEZ <- 9.95 :) update(MASS , cell_division_MASS_var);
when(: CLB2_KEZ -> 10, CLB2_KEZ <- 9.95 :) update(LTE1 , cell_division_LTE1_var);
when(: CLB2_KEZ -> 10, CLB2_KEZ <- 9.95 :) update(SPN, cell_division_SPN_var);

when(: ORI <- 0.1, ORI -> 1.0 :) update(MAD2 , start_S_MAD2_var);
when(: ORI <- 0.1, ORI -> 1.0 :) update(BUB2 , start_S_BUB2_var);

when(: SPN <- 0.08, SPN -> 1.0 :) update(MAD2 , spindle_checkpoint_MAD2_var);
when(: SPN <- 0.08, SPN -> 1.0 :) update(LTE1 , spindle_checkpoint_LTE1_var);
when(: SPN <- 0.08, SPN -> 1.0 :) update(BUB2 , spindle_checkpoint_BUB2_var);

when(: BUD -> 1.0e+170, BUD <- 10 :) update(A , useless);
when(: LTE1 -> 1.0e+170, LTE1 <- 10 :) update(A , useless);
when(: SPN -> 1.0e+170, SPN <- 10 :) update(A , useless);
when(: ORI -> 1.0e+170, ORI <- 10 :) update(A , useless);
when(: MAD2 -> 1.0e+170, MAD2 <- 10 :) update(A , useless);
when(: BUB2 -> 1.0e+170, BUB2 <- 10 :) update(A , useless);
when(: LTE1 -> 1.0e+170, LTE1 <- 10 :) update(A , useless);
when(: SBF -> 1.0e+170, SBF <- 10 :) update(A , useless);

let KILLER : bproc = #(k, KILLER) [ rep k!().nil ];

when(KILLER: |KILLER| < 4: inf) delete(2);
when(KILLER : MASS -> 10.48 : inf ) new(20);
when(: KILLER <- 2 , KILLER -> 10 : ) update(MASS, null_mass);

run init_PPX PPX || init_CDC20i CDC20i || init_C5 C5 || init_CDC6P CDC6P || init_C5P C5P ||
init_CDH1i CDH1i || init_CDC14 CDC14 || init_CDC15 CDC15 || init_CDH1 CDH1 || init_SIC1 SIC1 ||
init_SWI5 SWI5 || init_F5P F5P || init_C2 C2 || init_NET1 NET1 || init_SIC1P SIC1P ||
init_SWI5P SWI5P || init_IE IE || init_TEM1GTP TEM1GTP || init_CDC6 CDC6 || init_CLN2 CLN2 ||
init_RENTP RENTP || init_RENT RENT || init_CLB5 CLB5 || init_CDC20 CDC20 || init_TEM1GDP TEM1GDP ||
init_CLB2 CLB2 || init_C2P C2P || init_ESP1 ESP1 || init_CDC15i CDC15i || init_NET1P NET1P ||
init_F5 F5 || init_F2 F2 || init_PE PE || init_PDS1 PDS1 || init_IEP IEP || init_F2P F2P || 2 KILLER
```

A.1.2 .types file

```
{CDH1_4KILLER_ty, CDC14_OUT_ty,CDC20_out_ty, RENT_DEPH_ty, NET1P_DEPH_ty, CDC15_ACT_ty, CDC20_ACT_ty,
SWI5_DEPH_ty,SWI5_PH_ty, SIC1P_DEPH_ty, CDC6P_DEPH_ty, KILLER, C2_ty , C2P_ty , C5_ty , C5P_ty ,
CDC14_ty , CDC15_ty , CDC15I_ty , CDC20_ty , CDC20I_ty , CDC6_ty , CDC6P_ty , CDH1_ty , CDH1I_ty ,
CLB2_ty , CLB5_ty , CLN2_ty , ESP1_ty , F2_ty , F2P_ty , F5_ty , F5P_ty , IE_ty , IEP_ty , NET1_ty ,
NET1P_ty , PDS1_ty , PE_ty , PPX_ty , RENT_ty , RENTP_ty , SIC1_ty , SIC1P_ty , SWI5_ty , SWI5P_ty ,
TEM1GDP_ty , TEM1GTP_ty}
%%

{ (C2_ty, KILLER,  10000000000000000000000000), (C2P_ty , KILLER, 10000000000000000000000000),
(C5_ty , KILLER, 10000000000000000000000000), (C5P_ty ,  KILLER, 10000000000000000000000000),
(CDC14_ty ,  KILLER, 10000000000000000000000000), (CDC15_ty ,  KILLER, 10000000000000000000000000),
(CDC15I_ty ,  KILLER, 10000000000000000000000000), (CDC20_ty ,  KILLER, 10000000000000000000000000),
(CDC20I_ty ,  KILLER, 10000000000000000000000000), (CDC6_ty ,  KILLER, 10000000000000000000000000),
(CDC6P_ty ,  KILLER, 10000000000000000000000000), (CDH1_4KILLER_ty ,  KILLER, 10000000000000000000000000),
(CLB2_ty ,  KILLER, 10000000000000000000000000), (CLB5_ty ,  KILLER, 10000000000000000000000000),
(CLN2_ty ,  KILLER, 10000000000000000000000000), (ESP1_ty ,  KILLER, 10000000000000000000000000),
(F2_ty ,  KILLER, 10000000000000000000000000), (F2P_ty ,  KILLER, 10000000000000000000000000),
(F5_ty ,  KILLER, 10000000000000000000000000), (F5P_ty ,  KILLER, 10000000000000000000000000),
(IE_ty ,  KILLER, 10000000000000000000000000), (IEP_ty ,  KILLER, 10000000000000000000000000),
(NET1_ty ,  KILLER, 10000000000000000000000000), (NET1P_ty ,  KILLER, 10000000000000000000000000),
(PDS1_ty ,  KILLER, 10000000000000000000000000), (PE_ty ,  KILLER, 10000000000000000000000000),
(PPX_ty ,  KILLER, 10000000000000000000000000), (RENT_ty ,  KILLER, 10000000000000000000000000),
(RENTP_ty ,  KILLER, 10000000000000000000000000), (SIC1_ty ,  KILLER, 10000000000000000000000000),
(SIC1P_ty ,  KILLER, 10000000000000000000000000), (SWI5_ty ,  KILLER, 10000000000000000000000000),
(SWI5P_ty ,  KILLER, 10000000000000000000000000), (TEM1GDP_ty ,  KILLER, 10000000000000000000000000),
(TEM1GTP_ty , KILLER, 10000000000000000000000000),

(CLB2_ty, CDH1_ty, rate(Vdb2_1)),
(CLB2_ty, CDC20_out_ty, rate(Vdb2_2)),
(CLB5_ty, CDC20_out_ty, rate(Vdb5_1)),
(PDS1_ty, CDC20_out_ty, rate(Vdpds_1)),
(PDS1_ty, CDH1_ty, rate(Vdpds_2)),
(SIC1P_DEPH_ty, CDC14_OUT_ty, rate(Vppc1_1)),
(CDC6P_DEPH_ty, CDC14_OUT_ty, rate(Vppf6_1)),
(SWI5_DEPH_ty, CDC14_OUT_ty, rate(Activation_of_SWI5_1)),
(SWI5_PH_ty, CLB2_ty, rate(Inactivation_of_SWI5_1)),
(CDC20_ACT_ty, IEP_ty, rate(Activation_of_CDC20_1)),
(CDC15_ACT_ty, TEM1GDP_ty, rate(CDC15_activation_1)),
(CDC15_ACT_ty, TEM1GTP_ty, rate(CDC15_activation_2)),
(CDC15_ACT_ty, CDC14_OUT_ty, rate(CDC15_activation_3)),
(NET1P_DEPH_ty, PPX_ty, rate(Vppnet_1)),
(RENT_DEPH_ty, PPX_ty, rate(Vppnet_1))
}
```

A.1.3 .func file

```
let b0 : const = 0.054;          let bub2h : const = 1.0;       let bub2l : const = 0.2;
let C0 : const = 0.4;            let Dn3 : const = 1.0;         let ebudb5 : const = 1.0;
let ebudn2 : const = 0.25;       let ebudn3 : const = 0.05;     let ec1b2 : const = 0.45;
let ec1b5 : const = 0.1;         let ec1k2 : const = 0.03;      let ec1n2 : const = 0.06;
let ec1n3 : const = 0.3;         let ef6b2 : const = 0.55;      let ef6b5 : const = 0.1;
let ef6k2 : const = 0.03;        let ef6n2 : const = 0.06;      let ef6n3 : const = 0.3;
let eicdhb2 : const = 1.2;       let eicdhb5 : const = 8.0;     let eicdhn2 : const = 0.4;
let eicdhn3 : const = 0.25;      let eorib2 : const = 0.45;     let eorib5 : const = 0.9;
let esbfb5 : const = 2.0;        let esbfn2 : const = 2.0;      let esbfn3 : const = 10.0;
let J20ppx : const = 0.15;       let Jacdh : const = 0.03;      let Jaiep : const = 0.1;
let Jamcm : const = 0.1;         let Jasbf : const = 0.01;      let Jatem : const = 0.1;
let Jd2c1 : const = 0.05;        let Jd2f6 : const = 0.05;      let Jicdh : const = 0.03;
let Jiiep : const = 0.1;         let Jimcm : const = 0.1;       let Jisbf : const = 0.01;
let Jitem : const = 0.1;         let Jn3 : const = 6.0;         let Jpds : const = 0.04;
let Jspn : const = 0.14;         let ka15_p : const = 0.0020;   let ka15_p_p : const = 1.0;
let ka15p : const = 0.0010;      let ka20_p : const = 0.05;     let ka20_p_p : const = 0.2;
```

```
let kacdh_p : const = 0.01;      let kacdh_p_p : const = 0.8;   let kaiep : const = 0.1;
let kamcm : const = 1.0;         let kasb2 : const = 50.0;      let kasb5 : const = 50.0;
let kasbf : const = 0.38;        let kasesp : const = 50.0;     let kasf2 : const = 15.0;
let kasf5 : const = 0.01;        let kasrent : const = 200.0;   let kasrentp : const = 1.0;
let kaswi : const = 2.0;         let kd14 : const = 0.1;        let kd1c1 : const = 0.01;
let kd1f6 : const = 0.01;        let kd1pds_p : const = 0.01;   let kd20 : const = 0.3;
let kd2c1 : const = 1.0;         let kd2f6 : const = 1.0;       let kd2pds_p_p : const = 0.2;
let kd3c1 : const = 1.0;         let kd3f6 : const = 1.0;       let kd3pds_p_p : const = 0.04;
let kdb2_p : const = 0.0030;     let kdb2_p_p : const = 0.4;    let kdb2p : const = 0.15;
let kdb5_p : const = 0.01;       let kdb5_p_p : const = 0.16;   let kdbud : const = 0.06;
let kdcdh : const = 0.01;        let kdib2 : const = 0.05;      let kdib5 : const = 0.06;
let kdiesp : const = 0.5;        let kdif2 : const = 0.5;       let kdif5 : const = 0.01;
let kdirent : const = 1.0;       let kdirentp : const = 2.0;    let kdn2 : const = 0.12;
let kdnet : const = 0.03;        let kdori : const = 0.06;      let kdppx_p : const = 0.17;
let kdppx_p_p : const = 2.0;     let kdspn : const = 0.06;      let kdswi : const = 0.08;
let ki15 : const = 0.5;          let kicdh_p : const = 0.0010;  let kicdh_p_p : const = 0.08;
let kiiep : const = 0.15;        let kimcm : const = 0.15;      let kisbf_p : const = 0.6;
let kisbf_p_p : const = 8.0;     let kiswi : const = 0.05;      let kkpnet_p : const = 0.01;
let kkpnet_p_p : const = 0.6;    let kppc1 : const = 4.0;       let kppf6 : const = 4.0;
let kppnet_p : const = 0.05;     let kppnet_p_p : const = 3.0;  let ks14 : const = 0.2;
let ks1pds_p_p : const = 0.03;   let ks20_p : const = 0.0060;   let ks20_p_p : const = 0.6;
let ks2pds_p_p : const = 0.055;  let ksb2_p : const = 0.0010;   let ksb2_p_p : const = 0.04;
let ksb5_p : const = 8.0E-4;     let ksb5_p_p : const = 0.0050; let ksbud : const = 0.2;
let ksc1_p : const = 0.012;      let ksc1_p_p : const = 0.12;   let kscdh : const = 0.01;
let ksf6_p : const = 0.024;      let ksf6_p_p : const = 0.12;   let ksf6_p_p_p : const = 0.0040;
let ksn2_p : const = 0.0;        let ksn2_p_p : const = 0.15;   let ksnet : const = 0.084;
let ksori : const = 2.0;         let kspds_p : const = 0.0;     let ksppx : const = 0.1;
let ksspn : const = 0.1;         let ksswi_p : const = 0.0050;  let ksswi_p_p : const = 0.08;
let mad2h : const = 8.0;         let mad2l : const = 0.01;      let mdt : const = 90.0;
let lte1h: const = 1.0;          let lte1l : const = 0.1;       let ESP1T : const = 1.0;
let IET : const = 1.0;           let TEM1T : const = 1.0;       let CDC15T : const = 1.0;
let kez2 : const = 0.2;          let kez : const = 0.3;

let alpha : const = 0.0008166695;

let BCK2 : function = (b0*MASS);

let Visbf_1 : const = kisbf_p_p*alpha;
let Visbf : function = (kisbf_p+Visbf_1*|CLB2|);

let CLN3_1 : const = C0*Dn3;
let CLN3 : function = (CLN3_1*MASS/(Jn3+Dn3*MASS));

let Vppc1_1 : const = kppc1*alpha;
let Vppc1 : function = (Vppc1_1*|CDC14|);

let Vppf6_1 : const = kppf6*alpha;
let Vppf6 : function = (Vppf6_1*|CDC14|);

let Vaiep_1 : const = kaiep*alpha;
let Vaiep : function = (Vaiep_1*|CLB2|);

let Vacdh_1 : const = kacdh_p_p*alpha;
let Vacdh : function = (kacdh_p+Vacdh_1*|CDC14|);

let Vicdh_1 : const = kicdh_p_p*eicdhn3;
let Vicdh_2 : const = kicdh_p_p*eicdhn2*alpha;
let Vicdh_3 : const = kicdh_p_p*eicdhb5*alpha;
let Vicdh_4 : const = kicdh_p_p*eicdhb2*alpha;
let Vicdh : function = (kicdh_p+Vicdh_1*CLN3+Vicdh_2*|CLN2|+Vicdh_3*|CLB5|+Vicdh_4*|CLB2| );

let Vkpnet_1 : const = kkpnet_p_p*alpha;
let Vkpnet : function = ((kkpnet_p+Vkpnet_1*|CDC15|)*MASS);

let Vppnet_1 : const = kppnet_p_p*alpha;
```

```
let Vppnet : function = (kppnet_p+Vppnet_1 *|PPX|);

let Vasbf_1 : const = kasbf*esbfn2*alpha;
let Vasbf_2 : const = kasbf*esbfn3;
let Vasbf_3 : const = kasbf*esbfb5*alpha;
let Vasbf : function = (Vasbf_1*|CLN2|+Vasbf_2*(CLN3+BCK2)+Vasbf_3*|CLB5|);

let SBF : var = (2 * (Jisbf) * (Vasbf) / ((Visbf) - (Vasbf) + (Jasbf) * (Visbf) + (Jisbf) * (Vasbf)
+ sqrt( pow((Visbf) - (Vasbf) + (Jasbf) * (Visbf) + (Jisbf) * (Vasbf), 2) - 4 * ((Visbf) - (Vasbf)) * (Jisbf) * (Vasbf))));

let MCM1_1 : const = kamcm*alpha;
let MCM1 : function = (2 * (Jimcm) * (MCM1_1*|CLB2|) / ((kimcm) - (MCM1_1*|CLB2|) + (Jamcm) * (kimcm) +
(Jimcm) * (MCM1_1*|CLB2|) + sqrt( pow((kimcm) - (MCM1_1*|CLB2|) + (Jamcm) * (kimcm) + (Jimcm) * (MCM1_1*|CLB2|), 2)
- 4 * ((kimcm) - (MCM1_1*|CLB2|)) * (Jimcm) * (MCM1_1*|CLB2|))));

let Vd2c1_1 : const = kd2c1*ec1n3;
let Vd2c1_2 : const = kd2c1*ec1k2;
let Vd2c1_3 : const = kd2c1*ec1n2*alpha;
let Vd2c1_4 : const = kd2c1*ec1b5*alpha;
let Vd2c1_5 : const = kd2c1*ec1b2*alpha;
let Vd2c1 : function = Vd2c1_1*CLN3 + Vd2c1_2*BCK2 + Vd2c1_3 *|CLN2| + Vd2c1_4*|CLB5| + Vd2c1_5*|CLB2|;

let Vd2f6_1 : const = kd2f6*ef6n3;
let Vd2f6_2 : const = kd2f6*ef6k2;
let Vd2f6_3 : const = kd2f6*ef6n2*alpha;
let Vd2f6_4 : const = kd2f6*ef6b5*alpha;
let Vd2f6_5 : const = kd2f6*ef6b2*alpha;
let Vd2f6 : function = Vd2f6_1*CLN3 + Vd2f6_2*BCK2 + Vd2f6_3*|CLN2| + Vd2f6_4*|CLB5| + Vd2f6_5*|CLB2|;

let Vkpc1 : function = (kd1c1+Vd2c1/(Jd2c1+alpha*(|SIC1|+|C2|+|C5|+|SIC1P|+|C2P|+|C5P|)));

let Vkpf6 : function = (kd1f6+Vd2f6/(Jd2f6+alpha*(|CDC6|+|F2|+|F5|+|CDC6P|+|F2P|+|F5P|)));

let Vdb2_1 : const = kdb2_p_p*alpha;
let Vdb2_2 : const = kdb2p*alpha;
let Vdb2 : function = (kdb2_p+Vdb2_1*|CDH1|+Vdb2_2*|CDC20|);

let Vdb5_1 : const = kdb5_p_p*alpha;
let Vdb5 : function = (kdb5_p+Vdb5_1*|CDC20|);

let Vdpds_1 : const = kd2pds_p_p*alpha;
let Vdpds_2 : const = kd3pds_p_p*alpha;
let Vdpds : function = (kd1pds_p+Vdpds_1*|CDC20|+Vdpds_2*|CDH1|);

let Vdppx : function = (kdppx_p+kdppx_p_p*(J20ppx+alpha*|CDC20|)*Jpds/(Jpds+alpha*|PDS1|));

let CLB2T : function = alpha*(|CLB2|+|C2|+|C2P|+|F2|+|F2P|);
let CLB5T : function = alpha*(|CLB5|+|C5|+|C5P|+|F5|+|F5P|);
let CDC14T : function = alpha*(|CDC14|+|RENT|+|RENTP|);
let NET1T : function = alpha*(|NET1|+|NET1P|+|RENT|+|RENTP|);
let SIC1T : function = alpha*(|SIC1|+|C2|+|C5|+|SIC1P|+|C2P|+|C5P|);
let CDC6T : function = alpha*(|CDC6|+|F2|+|F5|+|CDC6P|+|F2P|+|F5P|);
let CKIT : function = (SIC1T+CDC6T);

let Synthesis_of_CLN2: function = ((ksn2_p+ksn2_p_p*SBF)*MASS)/alpha;
let Synthesis_of_CLB2: function = ((ksb2_p+ksb2_p_p*MCM1)*MASS)/alpha;
let Synthesis_of_CLB5: function = ((ksb5_p+ksb5_p_p*SBF)*MASS)/alpha;
let Synthesis_of_SIC1_1 : const = ksc1_p_p*alpha;
let Synthesis_of_SIC1: function = (ksc1_p+Synthesis_of_SIC1_1*|SWI5|)/alpha;
let Phosphorylation_of_SIC1: function = ((Vkpc1) * (|SIC1|));
let Assoc_of_CLB2_and_SIC1: function = ((kasb2) * (alpha*|CLB2|) * (|SIC1|));
let Dissoc_of_CLB2SIC1_complex: function = ((kdib2) * (|C2|));
let Assoc_of_CLB5_and_SIC1: function = ((kasb5) * (|CLB5|) * (alpha*|SIC1|));
let Dissoc_of_CLB5SIC1: function = ((kdib5) * (|C5|));
let Phosphorylation_of_C2: function = ((Vkpc1) * (|C2|));
```

```
let Dephosphorylation_of_C2P: function = ((Vppc1) * (|C2P|));
let Phosphorylation_of_C5: function = ((Vkpc1) * (|C5|));
let Dephosphorylation_of_C5P: function = ((Vppc1) * (|C5P|));
let Degradation_of_CLB2_in_C2: function = ((Vdb2) * (|C2|));
let Degradation_of_CLB5_in_C5: function = ((Vdb5) * (|C5|));
let Degradation_of_SIC1_in_C2P: function = ((kd3c1) * (|C2P|));
let Degradation_of_SIC1P_in_C5P_: function = ((kd3c1) * (|C5P|));
let Degradation_of_CLB2_in_C2P: function = ((Vdb2) * (|C2P|));
let Degradation_of_CLB5_in_C5P: function = ((Vdb5) * (|C5P|));
let CDC6_synthesis_1 : const = ksf6_p_p*alpha;
let CDC6_synthesis: function = (ksf6_p+CDC6_synthesis_1*|SWI5|+ksf6_p_p_p*SBF)/alpha;
let Phosphorylation_of_CDC6: function = ((Vkpf6) * (|CDC6|));
let CLB2CDC6_complex_formation: function = ((kasf2) * (|CLB2|) * (alpha*|CDC6|));
let CLB2CDC6_dissociation: function = ((kdif2) * (|F2|));
let CLB5CDC6_complex_formation: function = ((kasf5) * (|CLB5|) * (alpha*|CDC6|));
let CLB5CDC6_dissociation: function = ((kdif5) * (|F5|));
let F2_phosphorylation: function = ((Vkpf6) * (|F2|));
let F2P_dephosphorylation: function = ((Vppf6) * (|F2P|));
let F5_phosphorylation: function = ((Vkpf6) * (|F5|));
let F5P_dephosphorylation: function = ((Vppf6) * (|F5P|));
let CLB2_degradation_in_F2: function = ((Vdb2) * (|F2|));
let CLB5_degradation_in_F5: function = ((Vdb5) * (|F5|));
let CDC6_degradation_in_F2P: function = ((kd3f6) * (|F2P|));
let CDC6_degradation_in_F5P: function = ((kd3f6) * (|F5P|));
let CLB2_degradation_in_F2P: function = ((Vdb2) * (|F2P|));
let CLB5_degradation_in_F5P: function = ((Vdb5) * (|F5P|));
let Synthesis_of_SWI5: function = (ksswi_p+ksswi_p_p*MCM1)/alpha;
let Activation_of_SWI5_1 : const = kaswi*alpha;
let Inactivation_of_SWI5_1 : const = kiswi*alpha;
let Activation_of_IEP: function = ((|IE|) * (Vaiep) / ((Jaiep) + (alpha*|IE|)));
let Inactivation_1_IEP: function = ((kiiep) * (|IEP|) / ((Jiiep) + (alpha*|IEP|)));
let Synthesis_of_inactive_CDC20: function = (ks20_p+ks20_p_p*MCM1)/alpha;
let Activation_of_CDC20_1 : const = ka20_p_p*alpha;
let Inactivation_2_CDC20: function = ((MAD2) * (|CDC20|));
let CDH1_synthesis: function = (kscdh)/alpha;
let CDH1i_activation: function = ((|CDH1i|) * (Vacdh) / ((Jacdh) + (alpha*|CDH1i|)));
let Inactivation_3_CDH1: function = ((|CDH1|) * (Vicdh) / ((Jicdh) + (alpha*|CDH1|)));
let CDC14_synthesis: function = (ks14)/alpha;
let Assoc_with_NET1_to_form_RENT: function = ((kasrent) * (alpha*|CDC14|) * (|NET1|));
let Dissoc_from_RENT: function = ((kdirent) * (|RENT|));
let Assoc_with_NET1P_to_form_RENTP: function = ((kasrentp) * (alpha*|CDC14|) * (|NET1P|));
let Dissoc_from_RENP: function = ((kdirentp) * (|RENTP|));
let Net1_synthesis: function = (ksnet)/alpha;
let NET1_phosphorylation: function = ((Vkpnet) * (|NET1|));
let RENT_phosphorylation: function = ((Vkpnet) * (|RENT|));
let Degradation_of_NET1_in_RENT: function = ((kdnet) * (|RENT|));
let Degradation_of_NET1P_in_RENTP: function = ((kdnet) * (|RENTP|));
let Degradation_of_CDC14_in_RENT: function = ((kd14) * (|RENT|));
let Degradation_of_CDC14_in_RENTP: function = ((kd14) * (|RENTP|));
let TEM1_activation: function = ((|TEM1GDP|) * (LTE1) / ((Jatem) + (alpha*|TEM1GDP|)));
let inactivation_1_TEM1GTP: function = ((|TEM1GTP|) * (BUB2) / ((Jitem) + (alpha*|TEM1GTP|)));

let CDC15_activation_1 : const = ka15_p*alpha;
let CDC15_activation_2 : const = ka15_p_p*alpha;
let CDC15_activation_3 : const = ka15p*alpha;
let PPX_synthesis: function = (ksppx)/alpha;
let degradation_1_PPX: function = ((Vdppx) * (|PPX|));
let PDS1_synthesis: function = (kspds_p+ks1pds_p_p*SBF+ks2pds_p_p*MCM1)/alpha;
let Degradation_of_PDS1_in_PE: function = ((Vdpds) * (|PE|));
let Assoc_with_ESP1_to_form_PE: function = ((kasesp) * (alpha*|PDS1|) * (|ESP1|));
let Disso_from_PE: function = ((kdiesp) * (|PE|));

let init_C2 : const = 0.238404 / alpha;        let init_C2P : const = 0.024034 / alpha;
let init_C5 : const = 0.070081 / alpha;        let init_C5P : const = 0.006878 / alpha;
let init_CDC14 : const = 0.468344 / alpha;     let init_CDC15 : const = 0.656533 / alpha;
```

```
let init_CDC20 : const = 0.444296 / alpha;     let init_CDC20i : const = 1.472044 / alpha;
let init_CDC6 : const = 0.10758 / alpha;       let init_CDC6P : const = 0.015486 / alpha;
let init_CDH1 : const = 0.930499 / alpha;      let init_CDH1i : const = 0.0695 / alpha;
let init_CLB2 : const = 0.1469227 / alpha;     let init_CLB5 : const = 0.0518014 / alpha;
let init_CLN2 : const = 0.0652511 / alpha;     let init_ESP1 : const = 0.301313 / alpha;
let init_F2 : const = 0.236058 / alpha;        let init_F2P : const = 0.0273938 / alpha;
let init_F5 : const = 7.24E-5 / alpha;         let init_F5P : const = 7.91E-5 / alpha;
let init_IEP : const = 0.1015 / alpha;         let init_NET1 : const = 0.018645 / alpha;
let init_NET1P : const = 0.970271 / alpha;     let init_PDS1 : const = 0.025612 / alpha;
let init_PPX : const = 0.123179 / alpha;       let init_RENT : const = 1.04954 / alpha;
let init_RENTP : const = 0.6 / alpha;          let init_SIC1 : const = 0.0228776 / alpha;
let init_SIC1P : const = 0.00641 / alpha;      let init_SWI5 : const = 0.95 / alpha;
let init_SWI5P : const = 0.02 / alpha;         let init_TEM1GTP : const = 0.9 / alpha;

let init_CDC15i : const = CDC15T/alpha - init_CDC15;
let init_IE : const = IET/alpha - init_IEP;
let init_PE : const = ESP1T/alpha - init_ESP1;
let init_TEM1GDP : const =  TEM1T/alpha  - init_TEM1GTP;

let mu : function = (log(2)/mdt);
let D : function = (1.026/mu-32.0);
let F : function = (exp(-mu*D));

let Growth: function = (mu*MASS);
let DNA_synthesis: function = (ksori*(eorib5*alpha*|CLB5|+eorib2*alpha*|CLB2|));
let Negative_regulation_of_DNA_synthesis: function = (kdori * ORI);
let Budding: function = (ksbud*(ebudn2*alpha*|CLN2|+ebudn3*CLN3+ebudb5*alpha*|CLB5|));
let Negative_regulation_of_Cell_budding: function = (kdbud * BUD);
let Spindle_formation: function = (ksspn*alpha*|CLB2|/(Jspn+alpha*|CLB2|));
let Spindle_disassembly: function = (kdspn * SPN);

let init_ORI : const =  9.09E-4;
let init_BUD : const = 0.008473;
let init_SPN : const = 0.03;

let MASS(0.1) : var = Growth init 1.206019;
let ORI(0.0001) : var = DNA_synthesis - Negative_regulation_of_DNA_synthesis init 9.09E-4;
let BUD(0.0001) : var = Budding - Negative_regulation_of_Cell_budding init 0.008473;
let SPN(0.0001) : var = Spindle_formation - Spindle_disassembly init 0.03;
let MAD2(0.1) : var = 0.0000001 init 0.01;
let BUB2(0.1) : var = 0.0000001 init 0.2;
let LTE1(0.1) : var = 0.0000001 init 0.1;

let start_S_MAD2_var : function = mad2h;
let start_S_BUB2_var : function = bub2h;
let reset_ORI_ORI_var : function = 0;

let spindle_checkpoint_MAD2_var : function = mad2l;
let spindle_checkpoint_LTE1_var : function = lte1h;
let spindle_checkpoint_BUB2_var : function = bub2l;

let cell_division_MASS_var : function = F * MASS;
let cell_division_LTE1_var : function = lte1l;
let cell_division_BUD_var : function = 0;
let cell_division_SPN_var : function = 0;

let A : var = 1000; let useless : function = 2000;

let CLB2_KEZ : var = alpha*|CLB2| - kez + 10;
let CLB2_CLB5_KEZ2 : var = (alpha*|CLB2| + alpha*|CLB5|) - kez2 + 10;

let null_mass : function = 0;
```

A.2 Stochastic Simulation Algorithm

Chemical stochastic systems are usually represented by a chemical master equation (CME) that describes the time evolution of the probability distribution of the discrete molecule quantities (expressed by natural numbers). This evolution is a Continuous Time Markov Chain (CTMC). However because each state requires a separate variable, the system becomes quickly intractable as the number of chemical species and the number of each kind of molecule grows. It was therefore a major breakthrough when Gillespie developed exact stochastic simulation algorithms [74]. These algorithms are exact in the sense that they are equivalent to the results of the master equation, but instead of solving the probabilities for all trajectories simultaneously, the simulation method calculates single trajectories through the Monte Carlo sampling methods. Calculating many individual trajectories and studying the statistics of these trajectories can provide the same insights as obtained with the master equation. Gillespie proposes two mathematically equivalent methods: the direct method (DM) and the first reaction method (FRM). The algorithm is computationally expensive, so many modifications and adaptations exist: the efficient next reaction method by Gibson and Bruck [73] that achieves a significant reduction in complexity with respect to the Gillespie algorithms, tau-leaping [75] and hybrid techniques [157] where reactants in abundance are modelled with deterministic behaviour. The price to be paid when those more efficient techniques are used, is that the exactness of the theory behind the algorithm as it connects to the master equation is generally compromised, but they offer reasonable realizations for greatly improved timescales.
The FRM algorithm can be summarized by the following steps:

1. Initialize the number of molecules for each species and the initial time = 0;
2. Calculate the propensity value a_i for each $i \in 1, \ldots, m$ (where m is the total number of reactions in the system. Propensities represent the probability that a reaction R_j occurs in the next infinitesimal time interval and depend on the amount of reactants that are present in the system in that time instant);
3. For each $i \in 1, \ldots, m$ generate a putative time τ_i in accordance with an exponential distribution of parameter a_i;
4. Let τ_μ and μ be the fastest time and the corresponding reaction;
5. Update the number of molecules to reflect the execution of μ;
6. Set $t = t + \tau_\mu$;
7. Go back to Step 2 unless the number of all reactants is zero or the simulation time has been exceeded.

This method is used heavily in computational systems biology at the basis of almost all the stochastic simulators implemented for modelling biological systems with different computational languages.

A.3 Continuous Stochastic Logic and PRISM

Continuous Stochastic Logic (CSL) is a logical formalism for expressing properties of continuous time Markov chains and can be used in order to perform formal verification of systems with stochastic dynamics.

There are two types of formulae in CSL: *state formulae* and *path formulae*.

An atomic state formula express the fact that the system is in a specfic state of the Markov chain and, using the formal syntax that can be found in [4], state formulae can be connected by the classical logical operators in order to express more complex properties about the system.

A path formula expresses the probability that a specific sequence of state formulae are true along a path of the Markov chain defining the model. Path formulas are formulae of the form $f_1 U_{[a1,b1]}\ f_2 U_{[a2,b2]} \dots f_n$ where $f_1, f_2, \dots f_n$ are state formulae, and $a_1, b_1, \dots, a_{n-1}, b_{n-1}$ are not-negative rationals representing time intervals.

For example, the formula $\Psi = Pr_{>0.3}(aU[0.0, 4.0]\ b)$ is a state formula that formally expresses the property that with probability greater than 0.3, the system will remain in a state where the output is a before making a transition before 4.0 time units have elapsed to a state where the output is b.

The model checking problem for this logic was shown to be decidable and there are tools that allow the specification of a model and the verification of CSL formulae. One of the most widely used software for reaching this goal is the probabilistic model checker PRISM [89], a tool for formal modelling and analysis of systems which exhibit random or probabilistic behaviour. It supports three types of probabilistic models: discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs) and Markov decision processes (MDPs), plus extensions of these models with costs and rewards.

Models are described using a simple, state-based language and the tool provides support for automated analysis of a wide range of quantitative properties of these models. The property specification language incorporates the temporal logics PCTL, CSL, LTL and PCTL*, as well as extensions for quantitative specifications and costs/rewards.

A.4 Deterministic data (Chen model)

In the following tables we report the data about the deterministic solution of the model in [30]. For each of the 131 mutants we summarize the data that can be found in [146] and on the web-site `http://mpf.biol.vt.edu/research/budding_yeast_model/pp/`. For each viable mutant, we list the values of mass at division and length of G1-phase. For each inviable mutant, we list the arrest stage and the error type (see Section 4.4 for details).

Mutant	Viable (Inviable)	M.a.D. (Arr.st.)	Length G1 (Err.type)
WT galactose	viable	1.92	108.4
WT glucose	viable	2.62	36.0
1 cln1D cln2D	viable	6.39	51.0
2 GAL CLN2 cln1D cln2D	viable	1.43	59.4
3 cln1D cln2D sic1D	viable	3.59	6.4
4 cln1D cln2D cdh1D	viable	4.29	56.7
5 GAL CLN2 cln1D cln2D cdh1D	viable	0.61	153.7
6 cln3D	viable	3.77	55.2
7 GAL CLN3	viable	1.02	40.3
8 bck2D	viable	3.9	56.2
9 Multicopy BCK2	viable	2.54	32.3
10 cln1D cln2D bck2D	viable	8.45	51.2
11 cln3D bck2D	inviable	licensed	5
12 cln3D bck2D multicopy CLN2	inviable	licensed	5
13 cln3D bck2D GAL CLN2 cln1D cln2D	viable	1.7	46.8
14 cln3D bck2D sic1D	inviable	-	-
15 cln1D cln2D cln3D	inviable	licensed	5
16 cln1D cln2D cln3D GAL CLN2	viable	1.55	68.4
17 cln1D cln2D cln3D GAL CLN3	viable	1.4	43.6
18 cln1D cln2D cln3D sic1D	viable	5.8	5.8
19 cln1D cln2D cln3D cdh1D	inviable	separated	3
20 cln1D cln2D cln3D multi copy CLB5	viable	10.68	40.6
21 cln1D cln2D cln3D GAL CLB5	viable	3.86	119.0
22 cln1D cln2D cln3D multicopy BCK2	viable	8.75	49.8
23 cln1D cln2D cln3D GAL CLB2	inviable	licensed	5
24 cln1D cln2D cln3D apc ts	inviable	aligned	3
25 sic1D	viable	2.32	9.7
26 GAL SIC1	viable	2.81	144.8
27 GAL SIC1 dbD	inviable	licensed	5
28 GAL SIC1 cln1D cln2D	inviable	licensed	5
29 GAL SIC1 GAL CLN2 cln1D cln2D	viable	3.22	144.2
30 GAL SIC1 cln1D cln2D cdh1D	inviable	licensed	5

Mutant	Viable (Inviable)	M.a.D. (Arr.st.)	Length G1 (Err.type)
31 GAL SIC1 GAL CLN2 cln1D cln2D cdh1D	viable	2.8	155.4
32 cdh1D	viable	1.91	66.3
33 Cdh1 constitutively active	inviable	fired	3
34 sic1D cdh1D	viable	0.0	0.0
35 sic1D cdh1D GALL CDC20	viable	1.7	117.5
36 cdc6D2 49	viable	2.46	40.6
37 sic1D cdc6D2 49	viable	2.18	13.4
38 cdh1D cdc6D2 49	viable	1.96	60.2
39 sic1D cdc6D2 49 cdh1D	inviable	fired	3
40 sic1D cdc6D2 49 cdh1D GALL CDC20	viable	2.2	141.0
41 swi5D	viable	2.37	23.3
42 swi5D GAL CLB2	inviable	-	-
43 swi5D cdh1D	inviable	-	-
44 swi5D cdh1D GAL SIC1	viable	2.5	143.0
45 clb1D clb2D	inviable	fired	3
46 CLB1 clb2D	inviable	-	-
47 GAL CLB2	viable	1.57	162.0
48 Multicopy GAL CLB2	inviable	separated	3
49 CLB1 clb2D cdh1D	viable	0.0	0.0
50 CLB1 clb2D pds1D	viable	0.0	0.0
51 GAL CLB2 sic1D	inviable	separated	3
52 GAL CLB2 cdh1D	inviable	-	-
53 CLB2 dbD	inviable	separated	3
54 CLB2 dbD in galactose	inviable	separated	3
55 CLB2 dbD multicopy SIC1	viable	2.94	53.0
56 CLB2 dbD GAL SIC1	viable	3.08	119.8
57 CLB2 dbD multicopy CDC6	viable	0.0	0.0
58 CLB2 dbD cllb5D	inviable	separated	3
60 GAL CLB2 dbD	inviable	separated	3
61 cllb5D clb6D	viable	3.13	73.4
62 cln1D cln2D clb5D clb6D	inviable	licensed	5
63 GAL CLB5	viable	1.82	101.9
64 GAL CLB5 sic1D	inviable	unlicensed	1
65 GAL CLB5 cdh1D	viable	1.36	126.0
66 CLB5 dbD	viable	2.64	30.8
67 CLB5 dbD sic1D	inviable	-	-
68 CLB5 dbD pds1D	viable	2.24	38.1
69 CLB5 dbD pds1D cdc20D	inviable	separated	3
70 GAL CLB5 dbD	inviable	-	-

Mutant	Viable (Inviable)	M.a.D. (Arr.st.)	Length G1 (Err.type)
71 cdc20ts	inviable	aligned	3
72 cdc20D cllb5D	inviable	aligned	3
73 cdc20D pds1D	inviable	separated	3
74 cdc20D pds1D cllb5D	viable	3.02	77.8
75 GAL CDC20	inviable	fired	10
76 cdc20ts mad2D	inviable	aligned	3
77 cdc20ts bub2D	inviable	aligned	3
78 pds1D	viable	2.25	40.2
79 esp1ts	inviable	aligned	1
80 PDS1 dbD	inviable	aligned	1
81 GAL PDS1 dbD	inviable	aligned	1
82 GAL PDS1 dbD esp1ts	inviable	aligned	1
83 GAL ESP1 cdc20ts	inviable	separated	3
84 tem1D	inviable	separated	3
85 GAL TEM1	viable	2.08	112.2
86 tem1ts multicopy CDC15	viable	0.0	0.0
87 tem1ts GAL CDC15	viable	3.66	30.0
88 tem1D net1ts	viable	6.18	38.5
89 tem1D multicopy CDC14	viable	8.36	35.8
90 cdc15D	inviable	separated	3
91 Multicopy CDC15	viable	2.27	134.1
92 cdc15ts multicopy TEM1	inviable	-	-
93 cdc15D net1ts	viable	6.13	38.5
94 cdc15ts multicopy CDC14	viable	8.2	36.0
95 net1ts	viable	6.3	42.1
96 GAL NET1	inviable	separated	3
97 cdc14ts	inviable	separated	3
98 GAL CDC14	inviable	licensed	5
99 GAL NET1 GAL CDC14	viable	1.9	110.5
100 net1ts cdc20ts	inviable	aligned	1
101 cdc14ts GAL SIC1	inviable	-	-
103 cdc14ts sic1D	inviable	-	-
104 cdc14ts cdh1D	inviable	-	-
105 cdc14 ts GAL CLN2	inviable	-	-
106 TAB6 1	viable	3.35	42.0
107 TAB6 1 cdc15D	viable	3.83	34.5
108 TAB6 1 cllb5D clb6D	inviable	licensed	5
109 TAB6 1 CLB1 clb2D	viable	6.22	20.6
110 mad2D	viable	2.35	38.9

Mutant	Viable (Inviable)	M.a.D. (Arr.st.)	Length G1 (Err.type)
111 bub2D	viable	3.29	38.6
112 mad2D bub2D	viable	2.7	38.8
113 WT in nocodazole	inviable	-	-
114 mad2D in nocodazole	inviable	-	-
115 mad2D GAL TEM1 in nocodazole	inviable	-	-
116 mad2D pds1D in nocodazole	inviable	-	-
117 bub2D in nocodazole	inviable	-	-
118 bub2D pds1D in nocodazole	inviable	-	-
119 bub2D mad2D in nocodazole	inviable	-	-
120 pds1D in nocodazole	inviable	-	-
121 net1ts in nocodazole	inviable	-	-
122 APC A	viable	3.86	27.4
123 APC A cdh1D	inviable	separated	3
124 APC A cdh1D in galactose	inviable	-	-
125 APC A cdh1D multicopy SIC1	viable	0.0	0.0
126 APC A cdh1D GAL SIC1	viable	2.89	112.8
127 APC A cdh1D multicopy CDC6	viable	0.0	0.0
128 APC A cdh1D GAL CDC6	viable	2.63	57.2
129 APC A cdh1D multicopy CDC20	viable	1.96	65.1
130 APC A sic1D	viable	3.39	7.4
131 APC A GAL CLB2	inviable	separated	3